EMNLP 2011

Conference on Empirical Methods in Natural Language Processing

Proceedings of the Conference

July 27–31, 2011

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-937284-11-4 / 1-937284-11-5

General Chair's Preface

Welcome to EMNLP 2011, Conference on Empirical Methods in Natural Language Processing, a conference organised annually by SIGDAT, the Association for Computational Linguistics' special interest group on linguistic data and corpus-based approaches to NLP. This year the conference is held from July 27th to 29th at the John McIntyre Conference Centre, Edinburgh, UK.

This year EMNLP is, for the first time, not only a stand alone conference, but also an anchor conference to several workshops, that are held on July 30th and July 31st at the Informatics Forum, Edinburgh.

This year's conference continues the successful growing trend of previous years, attracting the largest number of papers to date for EMNLP and requiring a large organisational effort. I would like to thank all the people involved.

Regina Barzilay from MIT and Mark Johnson from Macquarie University chaired a large scientific programme committee and introduced several innovations in the submission, selection and final scheduling of papers.

Marie Candito, of Paris Diderot University, in collaboration with the ACL workshop chairs, selected the workshops, seven of which have been affiliated to EMNLP. They have received a large number of submissions and generated very interesting programmes.

Special thanks go to the publication chair Wanxiang Che, from Harbin Institute of Technology, who had to deal with EMNLP's now famous just-in-time schedule.

We are very grateful to Bonnie Webber and Miles Osborne, from the University of Edinburgh, for accepting the demanding task of organising the largest and most complex EMNLP ever and allowing us to hold this conference in such a remarkable city. Thanks also go to Francesco Figari the webmaster, and to all the student volunteers that make the conference possible.

Miles Osborne is also the contact person for an interesting affiliated event sponsored by Google and the Scottish and Informatics Computer Science Alliance: an Intense Summer School on Hadoop and Natural Language Processing, that will take place in Edinburgh for two days before the conference.

Thanks also to David Yarowsky and Ken Church from SIGDAT who provided much useful information from past conferences and Graeme Hirst and Priscilla Rasmussen from ACL for their help and advice.

This year's EMNLP's sponsors are Google, Yahoo and Textkernel. We thank them for their very welcome contributions, which were obtained by the efforts of the local organisers and of Massimilano Ciaramita (Google) and Stefan Riezler (Heidelberg University), our ACL sponsorship committee members for Europe.

Finally, and foremost, thank to all the authors and conference attendees that have made and will make this conference a success and source of inspiration.

Paola Merlo, University of Geneva
EMNLP 2011 General Chair

Program Chairs' Preface

Welcome to the EMNLP 2011 conference, hosted this year by the the extremely impressive University of Edinburgh.

EMNLP 2011 received 628 submissions, a new record for the conference. The program committee, consisted of 20 area chairs and 545 PC members from Asia, Europe, and North America, was able to accept 149 papers in total (for an acceptance rate of 23.7%). Among them, 94 (i.e., 63%) of the papers were accepted for oral presentations, and 54 (i.e., 36%) for poster presentations. Submissions from PC chairs were handled by the general chair in an off-line procedure that applied considerably stricter acceptance requirements than for ordinary submissions.

First and foremost, we would like to thank the authors who submitted their work to EMNLP 2011. The sheer number of submissions reflects how broad and active our field is. We are deeply indebted to the area chairs and the PC members for their hard work. They enabled us to make a wonderful program and to provide valuable feedback to the authors. We are extremely pleased that our invited speaker David McAllester has agreed to talk at EMNLP. Many thanks to local arrangements chairs, Bonnie Webber and Miles Osborne, who enabled the conference to be held in Edinburgh, one of the great intellectual cities of the world. We'd also like to thank the publications chair, Wanxiang Che, who put this volume together. We greatly benefited from advice from Hang Li and Lluís Màrquez who kindly shared with us their experience from EMNLP 2010. Special thanks to the general chair, Paola Merlo, who provided much valuable advice and assistance in the past months. We are grateful to David Yarowksy for assistance with a variety of aspects of the conference. Rich Gerber and the START team responded to our questions quickly, and helped us manage the large number of submissions smoothly; we would like to thank them as well.

This year's conference is innovative in several ways. The conference contains three additional plenary sessions compared to previous EMNLP conferences; these are used to highlight a diverse set of papers of interest to the entire EMNLP audience. We hope this will help counter the disciplinary fragmentation that some of us feel the standard multi-track conference structure encourages.

For the first time, submitted papers could be optionally accompanied by up to 10MB of supplementary material, which could consist of data, code, and text. Papers can reference the supplementary material in much the same way a paper might refer to software or a tech report available from the authors' web site (albeit without revealing the authors' identities). Reviewers were encouraged but not required to view the supplementary material.

A major challenge this year concerned undisclosed double submissions and plagiarism (especially self-plagiarism) involving papers accepted at other international conferences. We believe this is an issue that must addressed by the broader research community. Our community needs clear, well-publicised, standards on double and overlapping submissions, and also needs procedures for sharing information between relevant conferences to discourage double submission and self-plagiarism.

But we don't want to blow this issue out of proportion. By far the vast majority of EMNLP submissions described creative, innovative work that taken together substantially advances the field. The success of a conference is really a result of the great efforts of everybody involved. We hope that you enjoy this year's conference in the historic city of Edinburgh!

Regina Barzilay and Mark Johnson
EMNLP 2011 Program Co-Chairs

Program Co-chairs

Regina Barzilay, Massachusetts Institute of Technology
Mark Johnson, Macquarie University

Area Chairs

Emily M. Bender, University of Washington
Phil Blunsom, University of Oxford
Eugene Charniak, Brown University
Chris Dyer, Carnegie Mellon University
Jenny Finkel, Columbia University
Radu Florian, IBM Watson Research Center
Amir Globerson, The Hebrew University of Jerusalem
Keith Hall, Google Research
Minlie Huang, Tsinghua University
Katrin Kirchhoff, University of Washington
Mirella Lapata, University of Edinburgh
Chin-Yew Lin, Microsoft Research Asia
Bo Pang, Yahoo! Research
Stefan Riezler, University of Heidelberg
Ellen Riloff, University of Utah
Hinrich Schuetze, University of Stuttgart
Jian Su, Institute for Infocomm Research
Dekai Wu, Hong Kong University of Science and Technology
Luke Zettlemoyer, University of Washington

Local Arrangements Chair

Bonnie Webber, University of Edinburgh
Miles Osborne, University of Edinburgh

Publications Chair

Wanxiang Che, Harbin Institute of Technology

Reviewers

Omri Abend, Rodrigo Agerri, Eneko Agirre, Byung Gyu Ahn, Hua Ai, Enrique Alfonseca, Cecilia Ovesdotter Alm, Vamshi Ambati, Sophia Ananiadou, Shankar Ananthakrishnan, Marianna Apidianaki, Shilpa Arora, Abishek Arun, Nicholas Asher, Giuseppe Attardi, Necip Fazil Ayan, Collin Baker, Tim Baldwin, Carmen Banea, Ken Barker, Marco Baroni, Regina Barzilay, Mary Beckman, Beata Beigman Klebanov, Núria Bel, Anja Belz, Emily M. Bender, Luciana Benotti, Edward Benson, Jonathan Berant, Taylor Berg-Kirkpatrick, Sabine Bergler, Shane Bergsma,

Steven Bethard, Pushpak Bhattacharyya, Fadi Biadsy, Chris Biemann, Dan Bikel, Alexandra Birch, Steven Bird, John Blitzer, Michael Bloodgood, Phil Blunsom, Nate Bodenstab, Ondrej Bojar, Kalina Bontcheva, Johan Bos, Elizabeth Boschee, Jordan Boyd-Graber, S. R. K. Branavan, Thorsten Brants, Eric Breck, Chris Brew, Ted Briscoe, Chris Brockett, Sam Brody, Paul Buitelaar, Aljoscha Burchardt, Aoife Cahill, Chris Callison-Burch, Nicola Cancedda, Marie Candito, Yunbo Cao, Claire Cardie, Giuseppe Carenini, Andy Carlson, Marine Carpuat, Xavier Carreras, Francisco Casacuberta, Vittorio Castelli, Neus Català, Joyce Chai, Nate Chambers, Pi-Chuan Chang, Pi-Chuan Chang, Eugene Charniak, Wanxiang Che, Ciprian Chelba, Hsin-Hsi Chen, Colin Cherry, David Chiang, Laura Chiticariu, Yejin Choi, Christos Christodoulopoulos, Jennifer Chu-Carroll, Kenneth Church, Massimiliano Ciaramita, Philipp Cimiano, Alexander Clark, Peter Clark, Stephen Clark, James Clarke, Shay Cohen, Trevor Cohn, Nigel Collier, Michael Collins, Kevyn Collins-Thompson, Gao Cong, Marta Ruiz Costa-Jussa, Koby Crammer, Dan Cristea, Silviu Cucerzan, Aron Culotta, Walter Daelemans, Dipanjan Das, Sajib Dasgupta, Hal Daume, Dina Ddemner, Adrià de Gispert, Marie-Catherine de Marneffe, Vera Demberg, Steve DeNeefe, John DeNero, Pascal Denis, Anoop Deoras, Maarten de-Rijke, Mona Diab, Brian Dillon, Xiaowen Ding, Anna Divoli, Mark Dredze, Markus Dreyer, Rebecca Dridan, Gregory Druck, Amit Dubey, Chris Dyer, Koji Eguchi, Jacob Eisenstein, Jason Eisner, Michael Elhadad, Noemie Elhadad, Lloyd Elliott, Micha Elsner, Tanveer Faruquie, Benoit Favre, Christiane Fellbaum, Donghui Feng, Yansong Feng, Elena Filatova, Denis Filimonov, Katja Filippova, Jenny Finkel, Dan Flickinger, Radu Florian, George Foster, Jennifer Foster, Anette Frank, Martin Franz, Martin Franz, Alex Fraser, Marjorie Freedman, Michel Galley, Michael Gamon, Kuzman Ganchev, Kavita Ganesan, Juri Ganitkevitch, Wei Gao, Janfeng Gao, Konstantina Garoufi, Jan Gasthaus, Eric Gaussier, Dmitriy Genzel, Kallirroi Georgila, Daniel Gildea, Kevin Gimpel, Roxana Girju, Amir Globerson, Yoav Goldberg, Yoav Goldberg, Dan Goldwasser, Sharon Goldwater, José González-Brenes, julio gonzalo, Joao Graca, Agustin Gravano, Spence Green, Gregory Grefenstette, Ralph Grishman, Iryna Gurevych, Ben Hachey, Barry Haddow, Gholamreza Haffari, Aria Haghighi, Udo Hahn, Dilek Hakkani-Tur, Keith Hall, Ben Han, Sanda Harabagiu, Sasa Hasan, Xiadong He, Xiaodong He, John Hendernson, James Henderson, James Henderson, Iris Hendrickx, Aurelie Herbelot, Hugo Hernault, Graeme Hirst, Hieu Hoang, Julia Hockenmaier, Raphael Hoffmann, Kristy Hollingshead, Yunhua Hu, Yunfeng Huang, Liang Huang, Minlie Huang, Zhongqiang Huang, Nancy Ide, Gonzalo Iglesias, Ryu Iida, Piotr Indyk, Diana Inkpen, Kentaro Inui, Abe Ittycheriah, Tommi Jaakkola, Alpa Jain, Martin Jansche, Heng Ji, Kim Jin-Dong, Richard Johansson, Howard Johnson, Mark Johnson, Rie Johnson, Kristiina Jokinen, Rosie Jones, Dan Jurafsky, Min-Yen Kan, Pallika Kanani, Hiroshi Kanayama, Rohit Kate, Simon Keizer, Frank Keller, Chloé Kiddon, Su Nam Kim, Irwin King, Tracy Holloway King, Katrin Kirchhoff, Dietrich Klakow, Dan Klein, Alex Klementiev, Kevin Knight, Philipp Koehn, Ioannis Konstas, Terry Koo, Moshe Koppel, Zornitsa Kozareva, Marco Kuhlmann, Roland Kuhn, Jonas Kuhn, Seth Kulick, Ravi Kumar, Mirella Lapata, Alex Lascarides, Alon Lavie, Claudia Leacock, Matt Lease, Lillian Lee, Yoong Keok Lee, Gregor Leusch, Abby Levenberg, Effi Levi, Gina-Anne Levow, Roger Levy, Xiao Li, Wenjie Li, Fangtao Li, Hang Li, Linlin Li, Mu Li, Shoushan Li, Yunyao Li, Zhifei Li, Percy Liang, Marc Light, Chin-Yew Lin, DeKang Lin, Christina Lioma, Qiaoling Liu, Feifan Liu, Yan Liu, Ting Liu, Yang Liu, Ed Loper, Adam Lopez, John Lowe, Yue Lu, Xiaoqiang Luo, Bill MacCartney, Klaus Macherey, Wolfgang Macherey, Nitin Madnani, Bernardo Magnini, Francois Mairesse, Andreas Maletti, Suresh Manandhar, Gideon Mann, Chris Manning, Daniel Marcu, Montserrat Marimon, Lluís Màrquez, James Martin, Andre Martins, Sameer Maskey, Fabio Massimo Zanzotto, Yuji Matsumoto, Irina Matveeva,

Arne Mauser, Michael Maxwell, Andrew McCallum, Diana McCarthy, David McClosky, Ryan McDonald, Candace McKenna, Kathy McKeown, Paul McNamee, Dan Melamed, Chris Mellish, Arul Menezes, Donald Metzler, Adam Meyers, Haitao Mi, Rada Mihalcea, Tim Miller, Daniel Mills, Eleni Miltsakaki, David Mimno, Chang Ming-Wei, Shachar Mirkin, Vibhu Mittal, Sien Moens, Mary-Francine Moens, Saif Mohammad, Christof Monz, Taesun Moon, Raymond Mooney, Robert Moore, Roser Morante, Alessandro Moschitti, Rutu Mulkar-Mehta, Dragos Munteanu, Smaranda Muresan, Gabriel Murray, Tetsuji Nakagawa, Preslav Nakov, Ramesh Nallapati, Jason Naradowsky, Tahira Naseem, Vivi Nastase, Mark-Jan Nederhof, Ani Nenkova, Günter Neumann, Hermann Ney, Hwee Tou Ng, Vincent Ng, Malvina Nissim, Joakim Nivre, Tadashi Nomoto, Eric Nyberg, Brendan O'Connor, Stephan Oepen, Kemal Oflazer, Manabu Okumura, Sebastian Pado, Lluís Padró, John Paisley, Alexis Palmer, Sinno Jialin Pan, Bo Pang, Simone Paolo Ponzetto, Carolina Parada, Chris Parisien,Marius Pasca, Siddharth Patwardhan, Adam Pauls, Ted Pedersen, Anselmo Penas, Gerald Penn, Marco Pennacchiotti, Slav Petrov, Sasa Petrovic, Manfred Pinkal, Emily Pitler, Massimo Poesio, Thierry Poibeau, Joseph Polifroni, Hoifung Poon, Ana-Maria Popescu, Andrei Popescu-Belis, Richard Power, Sameer Pradhan, John Prager, Rashmi Prasad, Matthew Purver, James Pustejovsky, Chris Quirk, Stephan Raaijmakers, Dragomir Radev, Hema Raghavan, Ari Rappoport, Lev Ratinov, Sravana Reddy, Roi Reichart, Joseph Reisinger, David Reitter, Philip Resnik, Sebastian Riedel, Sebastian Riedel, Stefan Riezler, German Rigau, Ellen Riloff, Hae-Chang Rim, Laura Rimell, Eric Ringger, Alan Ritter, Barbara Rosario, Andrew Rosenberg, Paolo Rosso, Antti-Veikko Rosti, Benjamin Roth, Salim Roukos, Sasha Rush, Markus Saers, Kenji Sagae, Horacio Saggion, Mark Sammons, Anoop Sarkar, Yutaka Sasaki, Christina Sauper, David Schlangen, Nathan Schneider, Hinrich Schuetze, Karl Schultz, Lane Schwarz, Holger Schwenk, Frank Seide, Satoshi Sekine, Jean Senellart, Abhinav Sethy, Hendra Setiawan, Burr Settles, Zak Shafran, Kumar Shankar, Dan Shen, Libin Shen, Shuming Shi, Hideki Shima, Advaith Siddharthan, Khalil Sima'an, Michel Simard, Sameer Singh, Gabriel Skantze, Kevin Small, David Smith, Jason Smith, Noah Smith, Matthew Snover, Ben Snyder, Stephen Soderland, Swapna Somasundaran, David Sontag, Valentin Spitkovsky, Caroline Sporleder, Mark Steedman, Mark Stevenson, Suzanne Stevenson, Veselin Stoyanov, Michael Strube, Jian Su, Venkat Subramaniam, Maosong Sun, Yun-Hsuan Sung, Mihai Surdeanu, Jun Suzuki, Ben Swanson, Stan Szpakowicz, Hiroya Takamura, Jie Tang, Ben Taskar, Yee Whye Teh, Simone Teufel, Stefan Thater, Christoph Tillmann, Ivan Titov, Cigdem Toprak, Kristina Toutonova, Roy Tromble, George Tsatsaronis, Yoshimasa Tsuruoka, Gokhan Tur, Evelyne Tzoukermann, Raghavendra Udupa, Olga Uryupina, Jakob Uszkoreit, Benjamin Van Durme, Lucy Vanderwende, Paola Verlardi, Yannick Versley, Karin Verspoor, David Vilar, Andreas Vlachos, Stephan Vogel, Martin Volk, Xiaojun Wan, Stephen Wan, Chong Wang, Mengqiu Wang, Wen Wang, Wei Wang, Taro Watanabe, Bonnie Webber, David Weir, Ralph Weischedel, Ben Wellner, Scott Wen-tau Yih, Michael Wick, Jan Wiebe, Theresa Wilson, Shuly Wintner, kamfai wong, Kristian Woodsend, Dekai Wu, Yunqing Xia, Jinxi Xu, Peng Xu, Huichao Xue, Nianwen Xue, Alexander Yates, Hao Yu, Hong Yu, Yisong Yue, Francois Yvon, Richard Zens, Luke Zettlemoyer, Chengxiang Zhai, Joy Zhang, qi zhang, Tong Zhang, Hao Zhang, Hai Zhao, Ming Zhou, Jerry Zhu, Jun Zhu, Imed Zitouni, Andreas Zollmann, Geoff Zweig, Pierre Zweigenbaum

Best Reviewer Awards

Tim Baldwin, Ken Barker, Marco Baroni, Mary Beckman, Luciana Benotti, Steven Bethard, Chris Biemann, Phil Blunsom, Chris Brockett, Sam Brody, Nate Chambers, Jennifer Chu-Carroll,

Peter Clark, Dina Ddemner, Markus Dreyer, Rebecca Dridan, Benjamin Van Durme, Chris Dyer, Michael Elhadad, Donghui Feng, Katja Filippova, Anette Frank, Kallirroi Georgila, Kevin Gimpel, Ben Hachey, John Henderson, Graeme Hirst, Dan Jurafsky, Min-Yen Kan, Chloe Kiddon, Tracy King, Kevin Knight, Terry Koo, Roger Levy, Bill MacCartney, Lluis Marquez, Andre Martins, Ryan McDonald, David Minmo, Saif Mohammad, Raymond Mooney, Ani Nenkova, Günter Neumann, Vincent Ng, Tadashi Nomoto, Alexis Palmer, Ted Pedersen, Sasa Petrovic, Emily Pitler, Hoifung Poon, Ari Rappoport, Eric Ringer, Alan Ritter, Mark Sammons, Yutaka Sasaki, Kevin Small, Noah Smith, Suzanne Stevenson, Yee Whye Teh, Simone Teufel, Raghavendra Udupa, Paola Verlardi, Andreas Vlachos, Stephen Wan, Alexander Yates, Scott Wen-tau Yih, Fabio Massimo Zanzotto, Pierre Zweigenbaum

Table of Contents

Conference Program

Wednesday, July 27, 2011

9:00–10:20 Plenary session: Opening and Invited Talk
Chair: Paola Merlo
Invited Talk: Object Detection Grammars (David McAllester)

10:20–11:00 Coffee break

Session 1: Plenary session
Chair: Jason Eisner

11:00–11:25 *Fast and Robust Joint Models for Biomedical Event Extraction*
Sebastian Riedel and Andrew McCallum

11:25–11:50 *Predicting Thread Discourse Structure over Technical Web Forums*
Li Wang, Marco Lui, Su Nam Kim, Joakim Nivre and Timothy Baldwin

11:50–12:15 *Exact Decoding of Phrase-Based Translation Models through Lagrangian Relaxation*
Yin-Wen Chang and Michael Collins

12:15–12:40 *Optimal Search for Minimum Error Rate Training*
Michel Galley and Chris Quirk

12:40–14:10 Lunch

Session 2A: Syntax and Parsing
Chair: Stephen Clark

14:10–14:35 *Unsupervised Structure Prediction with Non-Parallel Multilingual Guidance*
Shay B. Cohen, Dipanjan Das and Noah A. Smith

14:35–15:00 *Multi-Source Transfer of Delexicalized Dependency Parsers*
Ryan McDonald, Slav Petrov and Keith Hall

15:00–15:25 *SMT Helps Bitext Dependency Parsing*
Wenliang Chen, Jun'ichi Kazama, Min Zhang, Yoshimasa Tsuruoka, Yujie Zhang, Yiou Wang, Kentaro Torisawa and Haizhou Li

15:25–15:50 *Accurate Parsing with Compact Tree-Substitution Grammars: Double-DOP*
Federico Sangati and Willem Zuidema

Wednesday, July 27, 2011 (continued)

Session 2B: Semantics
Chair: Mirella Lapata

14:10–14:35 *A Generate and Rank Approach to Sentence Paraphrasing*
Prodromos Malakasiotis and Ion Androutsopoulos

14:35–15:00 *Correcting Semantic Collocation Errors with L1-induced Paraphrases*
Daniel Dahlmeier and Hwee Tou Ng

15:00–15:25 *Class Label Enhancement via Related Instances*
Zornitsa Kozareva, Konstantin Voevodski and Shanghua Teng

15:25–15:50 *A Joint Model for Extended Semantic Role Labeling*
Vivek Srikumar and Dan Roth

Session 2C: Sentiment Analysis and Opinion Mining
Chair: Bo Pang

14:10–14:35 *Domain-Assisted Product Aspect Hierarchy Generation: Towards Hierarchical Organization of Unstructured Consumer Reviews*
Jianxing Yu, Zheng-Jun Zha, Meng Wang, Kai Wang and Tat-Seng Chua

14:35–15:00 *Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions*
Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng and Christopher D. Manning

15:00–15:25 *Unsupervised Discovery of Discourse Relations for Eliminating Intra-sentence Polarity Ambiguities*
Lanjun Zhou, Binyang Li, Wei Gao, Zhongyu Wei and Kam-Fai Wong

15:25–15:50 *Compositional Matrix-Space Models for Sentiment Analysis*
Ainur Yessenalina and Claire Cardie

Wednesday, July 27, 2011 (continued)

Session 3A: Machine Translation
Chair: Phil Blunsom

16:20–16:45 *Training a Parser for Machine Translation Reordering*
Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno and Hideto Kazawa

16:45–17:10 *Inducing Sentence Structure from Parallel Corpora for Reordering*
John DeNero and Jakob Uszkoreit

17:10–17:35 *Augmenting String-to-Tree Translation Models with Fuzzy Use of Source-side Syntax*
Jiajun Zhang, Feifei Zhai and Chengqing Zong

17:35–18:00 *A novel dependency-to-string model for statistical machine translation*
Jun Xie, Haitao Mi and Qun Liu

Session 3B: NLP related Machine Learning
Chair: David Smith

16:20–16:45 *Bayesian Checking for Topic Models*
David Mimno and David Blei

16:45–17:10 *Dual Decomposition with Many Overlapping Components*
Andre Martins, Noah Smith, Mario Figueiredo and Pedro Aguiar

17:10–17:35 *Approximate Scalable Bounded Space Sketch for Large Data NLP*
Amit Goyal and Hal Daume III

17:35–18:00 *Optimizing Semantic Coherence in Topic Models*
David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders and Andrew McCallum

Wednesday, July 27, 2011 (continued)

Session 3C: Discourse Dialogue and Pragmatics
Chair: Oliver Lemon

16:20–16:45 *A Weakly-supervised Approach to Argumentative Zoning of Scientific Documents*
Yufan Guo, Anna Korhonen and Thierry Poibeau

16:45–17:10 *Linear Text Segmentation Using Affinity Propagation*
Anna Kazantseva and Stan Szpakowicz

17:10–17:35 *Minimally Supervised Event Causality Identification*
Quang Do, Yee Seng Chan and Dan Roth

17:35–18:00 *A Model of Discourse Predictions in Human Sentence Processing*
Amit Dubey, Frank Keller and Patrick Sturt

Thursday, July 28, 2011

Session 4: Plenary session
Chair: Michael Collins

9:05–9:30 *Simple Effective Decipherment via Combinatorial Optimization*
Taylor Berg-Kirkpatrick and Dan Klein

9:30–9:55 *Universal Morphological Analysis using Structured Nearest Neighbor Prediction*
Young-Bum Kim, João Graça and Benjamin Snyder

9:55–10:20 *Training a Log-Linear Parser with Loss Functions via Softmax-Margin*
Michael Auli and Adam Lopez

10:20–11:00 Coffee break

Thursday, July 28, 2011 (continued)

Session 5A: Machine Translation
Chair: Philipp Koehn

11:00–11:25 *Large-Scale Cognate Recovery*
David Hall and Dan Klein

11:25–11:50 *Domain Adaptation via Pseudo In-Domain Data Selection*
Amittai Axelrod, Xiaodong He and Jianfeng Gao

11:50–12:15 *Language Models for Machine Translation: Original vs. Translated Texts*
Gennadi Lembersky, Noam Ordan and Shuly Wintner

12:15–12:40 *Better Evaluation Metrics Lead to Better Machine Translation*
Chang Liu, Daniel Dahlmeier and Hwee Tou Ng

Session 5B: Syntax and Parsing
Chair: Ryan McDonald

11:00–11:25 *Evaluating Dependency Parsing: Robust and Heuristics-Free Cross-Annotation Evaluation*
Reut Tsarfaty, Joakim Nivre and Evelina Andersson

11:25–11:50 *Parser Evaluation over Local and Non-Local Deep Dependencies in a Large Corpus*
Emily M. Bender, Dan Flickinger, Stephan Oepen and Yi Zhang

11:50–12:15 *Learning to Simplify Sentences with Quasi-Synchronous Grammar and Integer Programming*
Kristian Woodsend and Mirella Lapata

12:15–12:40 *Bootstrapping Semantic Parsers from Conversations*
Yoav Artzi and Luke Zettlemoyer

Thursday, July 28, 2011 (continued)

Session 5C: Summarization and Generation
Chair: Johanna Moore

11:00–11:25 *Timeline Generation through Evolutionary Trans-Temporal Summarization*
Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li and Yan Zhang

11:25–11:50 *Corpus-Guided Sentence Generation of Natural Images*
Yezhou Yang, Ching Teo, Hal Daume III and Yiannis Aloimonos

11:50–12:15 *Corroborating Text Evaluation Results with Heterogeneous Measures*
Enrique Amigó, Julio Gonzalo, Jesus Gimenez and Felisa Verdejo

12:15–12:40 *Ranking Human and Machine Summarization Systems*
Peter Rankel, John Conroy, Eric Slud and Dianne O'Leary

12:40–14:10 Lunch

Session 6A: Machine Translation
Chair: Stefan Riezler

14:10–14:35 *Quasi-Synchronous Phrase Dependency Grammars for Machine Translation*
Kevin Gimpel and Noah A. Smith

14:35–15:00 *A Word Reordering Model for Improved Machine Translation*
Karthik Visweswariah, Rajakrishnan Rajkumar, Ankur Gandhe, Ananthakrishnan Ramanathan and Jiri Navratil

15:00–15:25 *Feature-Rich Language-Independent Syntax-Based Alignment for Statistical Machine Translation*
Jason Riesa, Ann Irvine and Daniel Marcu

15:25–15:50 *Efficient retrieval of tree translation examples for Syntax-Based Machine Translation*
Fabien Cromieres and Sadao Kurohashi

Thursday, July 28, 2011 (continued)

Session 6B: Semantics
Chair: Hwee Tou Ng

14:10–14:35 *A generative model for unsupervised discovery of relations and argument classes from clinical texts*
Bryan Rink and Sanda Harabagiu

14:35–15:00 *Random Walk Inference and Learning in A Large Scale Knowledge Base*
Ni Lao, Tom Mitchell and William W. Cohen

15:00–15:25 *Exploring Supervised LDA Models for Assigning Attributes to Adjective-Noun Phrases*
Matthias Hartung and Anette Frank

15:25–15:50 *Semantic Topic Models: Combining Word Distributional Statistics and Dictionary Definitions*
Weiwei Guo and Mona Diab

Session 6C: Sentiment Analysis and Opinion Mining
Chair: Benjamin Snyder

14:10–14:35 *Cooooooooooooooollllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs*
Samuel Brody and Nicholas Diakopoulos

14:35–15:00 *Personalized Recommendation of User Comments via Factor Models*
Deepak Agarwal, Bee-Chung Chen and Bo Pang

15:00–15:25 *Data-Driven Response Generation in Social Media*
Alan Ritter, Colin Cherry and William B. Dolan

15:25–15:50 *Predicting a Scientific Community's Response to an Article*
Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R. Routledge and Noah A. Smith

15:50–16:20 Coffee break

Thursday, July 28, 2011 (continued)

Session 7A: Phonology Morphology Tagging Chunking and Segmentation
Chair: Noah Smith

16:20–16:45 *Non-parametric Bayesian Segmentation of Japanese Noun Phrases*
Yugo Murawaki and Sadao Kurohashi

16:45–17:10 *Discovering Morphological Paradigms from Plain Text Using a Dirichlet Process Mixture Model*
Markus Dreyer and Jason Eisner

17:10–17:35 *Multilayer Sequence Labeling*
Ai Azuma and Yuji Matsumoto

17:35–18:00 *A Bayesian Mixture Model for PoS Induction Using Multiple Features*
Christos Christodoulopoulos, Sharon Goldwater and Mark Steedman

Session 7B: Semantics
Chair: Mark Stevenson

16:20–16:45 *Large-Scale Noun Compound Interpretation Using Bootstrapping and the Web as a Corpus*
Su Nam Kim and Preslav Nakov

16:45–17:10 *Linguistic Redundancy in Twitter*
Fabio Massimo Zanzotto, Marco Pennaccchiotti and Kostas Tsioutsiouliklis

17:10–17:35 *Divide and Conquer: Crowdsourcing the Creation of Cross-Lingual Textual Entailment Corpora*
Matteo Negri, Luisa Bentivogli, Yashar Mehdad, Danilo Giampiccolo and Alessandro Marchetti

17:35–18:00 *Literal and Metaphorical Sense Identification through Concrete and Abstract Context*
Peter Turney, Yair Neuman, Dan Assaf and Yohai Cohen

Thursday, July 28, 2011 (continued)

Session 7C: Spoken Language + IR
Chair: Steve Renals

16:20–16:45 *Syntactic Decision Tree LMs: Random Selection or Intelligent Design?*
Denis Filimonov and Mary Harper

16:45–17:10 *The Imagination of Crowds: Conversational AAC Language Modeling using Crowdsourcing and Large Data Sources*
Keith Vertanen and Per Ola Kristensson

17:10–17:35 *Using Syntactic and Semantic Structural Kernels for Classifying Definition Questions in Jeopardy!*
Alessandro Moschitti, Jennifer Chu-carroll, Siddharth Patwardhan, James Fan and Giuseppe Riccardi

17:35–18:00 *Multiword Expression Identification with Tree Substitution Grammars: A Parsing tour de force with French*
Spence Green, Marie-Catherine de Marneffe, John Bauer and Christopher D. Manning

18:30–21:30 Poster session and Reception

Modelling Discourse Relations for Arabic
Amal Al-Saif and Katja Markert

Classifying Sentences as Speech Acts in Message Board Posts
Ashequl Qadir and Ellen Riloff

Learning Local Content Shift Detectors from Document-level Information
Richard Farkas

Collaborative Ranking: A Case Study on Entity Linking
Zheng Chen and Heng Ji

Robust Disambiguation of Named Entities in Text
Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater and Gerhard Weikum

A Cascaded Classification Approach to Semantic Head Recognition
Lukas Michelbacher, Alok Kothari, Martin Forst, Christina Lioma and Hinrich Schütze

Linking Entities to a Knowledge Base with Query Expansion
Swapna Gottipati and Jing Jiang

Unsupervised Information Extraction with Distributional Prior Knowledge
Cane Wing-ki Leung, Jing Jiang, Kian Ming A. Chai, Hai Leong Chieu and Loo-Nin Teow

Relation Acquisition using Word Classes and Partial Patterns
Stijn De Saeger, Kentaro Torisawa, Masaaki Tsuchida, Jun'ichi Kazama, Chikara Hashimoto, Ichiro Yamada, Jong Hoon Oh, Istvan Varga and Yulan Yan

Identification of Multi-word Expressions by Combining Multiple Linguistic Information Sources
Yulia Tsvetkov and Shuly Wintner

Analyzing Methods for Improving Precision of Pivot Based Bilingual Dictionaries
Xabier Saralegi, Iker Manterola and Iñaki San Vicente

Soft Dependency Constraints for Reordering in Hierarchical Phrase-Based Translation
Yang Gao, Philipp Koehn and Alexandra Birch

Statistical Machine Translation with Local Language Models
Christof Monz

Fast Generation of Translation Forest for Large-Scale SMT Discriminative Training
Xinyan Xiao, Yang Liu, Qun Liu and Shouxun Lin

A Correction Model for Word Alignments
J. Scott McCarley, Abraham Ittycheriah, Salim Roukos, Bing Xiang and Jian-ming Xu

Heuristic Search for Non-Bottom-Up Tree Structure Prediction
Andrea Gesmundo and James Henderson

Cache-based Document-level Statistical Machine Translation
Zhengxian Gong, Min Zhang and Guodong Zhou

Minimum Imputed-Risk: Unsupervised Discriminative Training for Machine Translation
Zhifei Li, Ziyuan Wang, Jason Eisner, Sanjeev Khudanpur and Brian Roark

Improving Bilingual Projections via Sparse Covariance Matrices
Jagadeesh Jagarlamudi, Raghavendra Udupa, Hal Daume III and Abhijit Bhole

Thursday, July 28, 2011 (continued)

Entire Relaxation Path for Maximum Entropy Problems
Moshe Dubiner and Yoram Singer

A Non-negative Matrix Factorization Based Approach for Active Dual Supervision from Document and Word Labels
Chao Shen and Tao Li

Splitting Noun Compounds via Monolingual and Bilingual Paraphrasing: A Study on Japanese Katakana Words
Nobuhiro Kaji and Masaru Kitsuregawa

Enhancing Chinese Word Segmentation Using Unlabeled Data
Weiwei Sun and Jia Xu

Unsupervised Learning of Selectional Restrictions and Detection of Argument Coercions
Kirk Roberts and Sanda Harabagiu

Harnessing different knowledge sources to measure semantic relatedness under a uniform model
Ziqi Zhang, Anna Lisa Gentile and Fabio Ciravegna

Refining the Notions of Depth and Density in WordNet-based Semantic Similarity Measures
Tong Wang and Graeme Hirst

Latent Vector Weighting for Word Meaning in Context
Tim Van de Cruys, Thierry Poibeau and Anna Korhonen

Hierarchical Verb Clustering Using Graph Factorization
Lin Sun and Anna Korhonen

Structured Lexical Similarity via Convolution Kernels on Dependency Trees
Danilo Croce, Alessandro Moschitti and Roberto Basili

Probabilistic models of similarity in syntactic context
Diarmuid Ó Séaghdha and Anna Korhonen

Lexical Co-occurrence, Statistical Significance, and Word Association
Dipak L. Chaudhari, Om P. Damani and Srivatsan Laxman

Learning the Information Status of Noun Phrases in Spoken Dialogues
Altaf Rahman and Vincent Ng

Harnessing WordNet Senses for Supervised Sentiment Classification
Balamurali AR, Aditya Joshi and Pushpak Bhattacharyya

Learning General Connotation of Words using Graph-based Algorithms
Song Feng, Ritwik Bose and Yejin Choi

Hypotheses Selection Criteria in a Reranking Framework for Spoken Language Understanding
Marco Dinarelli and Sophie Rosset

A Fast Re-scoring Strategy to Capture Long-Distance Dependencies
Anoop Deoras, Tomas Mikolov and Kenneth Church

Efficient Subsampling for Training Complex Language Models
Puyang Xu, Asela Gunawardana and Sanjeev Khudanpur

Generating Aspect-oriented Multi-Document Summarization with Event-aspect model
Peng Li, Yinglin Wang, Wei Gao and Jing Jiang

Syntax-Based Grammaticality Improvement using CCG and Guided Search
Yue Zhang and Stephen Clark

Generating Subsequent Reference in Shared Visual Scenes: Computation vs Re-Use
Jette Viethen, Robert Dale and Markus Guhe

Learning Sentential Paraphrases from Bilingual Parallel Corpora for Text-to-Text Generation
Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles and Benjamin Van Durme

Joint Models for Chinese POS Tagging and Dependency Parsing
Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen and Haizhou Li

Relaxed Cross-lingual Projection of Constituent Syntax
Wenbin Jiang, Qun Liu and Yajuan Lv

Thursday, July 28, 2011 (continued)

Computing Logical Form on Regulatory Texts
Nikhil Dinesh, Aravind Joshi and Insup Lee

Computation of Infix Probabilities for Probabilistic Context-Free Grammars
Mark-Jan Nederhof and Giorgio Satta

Parse Correction with Specialized Models for Difficult Attachment Types
Enrique Henestroza Anguiano and Marie Candito

Exact Inference for Generative Probabilistic Non-Projective Dependency Parsing
Shay B. Cohen, Carlos Gómez-Rodríguez and Giorgio Satta

Semi-supervised CCG Lexicon Extension
Emily Thomforde and Mark Steedman

A Fast, Accurate, Non-Projective, Semantically-Enriched Parser
Stephen Tratz and Eduard Hovy

Lateen EM: Unsupervised Training with Multiple Objectives, Applied to Dependency Grammar Induction
Valentin I. Spitkovsky, Hiyan Alshawi and Daniel Jurafsky

Unsupervised Dependency Parsing without Gold Part-of-Speech Tags
Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang and Daniel Jurafsky

Exploiting Syntactic and Distributional Information for Spelling Correction with Web-Scale N-gram Models
Wei Xu, Joel Tetreault, Martin Chodorow, Ralph Grishman and Le Zhao

Discriminating Gender on Twitter
John D. Burger, John Henderson, George Kim and Guido Zarrella

Identifying and Following Expert Investors in Stock Microblogs
Roy Bar-Haim, Elad Dinur, Ronen Feldman, Moshe Fresko and Guy Goldstein

Friday, July 29, 2011

Session 8: Plenary session
Chair: Shuly Wintner

9:05–9:30 *Unsupervised Semantic Role Induction with Graph Partitioning*
Joel Lang and Mirella Lapata

9:30–9:55 *Structural Opinion Mining for Graph-based Sentiment Representation*
Yuanbin Wu, Qi Zhang, Xuanjing Huang and Lide Wu

9:55–10:20 *Summarize What You Are Interested In: An Optimization Framework for Interactive Personalized Summarization*
Rui Yan, Jian-Yun Nie and Xiaoming Li

10:20–10:50 Coffee break

Session 9A: Machine Translation
Chair: John DeNero

10:50–11:15 *Tuning as Ranking*
Mark Hopkins and Jonathan May

11:15–11:40 *Watermarking the Outputs of Structured Prediction with an application in Statistical Machine Translation.*
Ashish Venugopal, Jakob Uszkoreit, David Talbot, Franz Och and Juri Ganitkevitch

11:40–12:05 *Hierarchical Phrase-based Translation Representations*
Gonzalo Iglesias, Cyril Allauzen, William Byrne, Adrià de Gispert and Michael Riley

12:05–12:30 *Improved Transliteration Mining Using Graph Reinforcement*
Ali El Kahki, Kareem Darwish, Ahmed Saad El Din, Mohamed Abd El-Wahab, Ahmed Hefny and Waleed Ammar

Friday, July 29, 2011 (continued)

Session 9B: Semantics
Chair: Peter Turney

10:50–11:15 *Experimental Support for a Categorical Compositional Distributional Model of Meaning*
Edward Grefenstette and Mehrnoosh Sadrzadeh

11:15–11:40 *Cross-Cutting Models of Lexical Semantics*
Joseph Reisinger and Raymond Mooney

11:40–12:05 *Reducing Grounded Learning Tasks To Grammatical Inference*
Benjamin Börschinger, Bevan K. Jones and Mark Johnson

12:05–12:30 *Relation Extraction with Relation Topics*
Chang Wang, James Fan, Aditya Kalyanpur and David Gondek

Session 9C: Information Extraction
Chair: Alessandro Moschitti

10:50–11:15 *Extreme Extraction – Machine Reading in a Week*
Marjorie Freedman, Lance Ramshaw, Elizabeth Boschee, Ryan Gabbard, Gary Kratkiewicz, Nicolas Ward and Ralph Weischedel

11:15–11:40 *Discovering Relations between Noun Categories*
Thahir Mohamed, Estevam Hruschka and Tom Mitchell

11:40–12:05 *Structured Relation Discovery using Generative Models*
Limin Yao, Aria Haghighi, Sebastian Riedel and Andrew McCallum

12:05–12:30 *Closing the Loop: Fast, Interactive Semi-Supervised Annotation With Queries on Features and Instances*
Burr Settles

12:30–13:00 SIGDAT Business Meeting

13:00–14:10 Lunch

Friday, July 29, 2011 (continued)

Session 10A: Syntax and Parsing
Chair: Mark Steedman

14:10–14:35 *Third-order Variational Reranking on Packed-Shared Dependency Forests*
Katsuhiko Hayashi, Taro Watanabe, Masayuki Asahara and Yuji Matsumoto

14:35–15:00 *Training dependency parsers by jointly optimizing multiple objectives*
Keith Hall, Ryan McDonald, Jason Katz-Brown and Michael Ringgaard

15:00–15:25 *Structured Sparsity in Structured Prediction*
Andre Martins, Noah Smith, Mario Figueiredo and Pedro Aguiar

15:25–15:50 *Lexical Generalization in CCG Grammar Induction for Semantic Parsing*
Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater and Mark Steedman

Session 10B: Information Extraction
Chair: Sebastian Riedel

14:10–14:35 *Named Entity Recognition in Tweets: An Experimental Study*
Alan Ritter, Sam Clark, Mausam and Oren Etzioni

14:35–15:00 *Identifying Relations for Open Information Extraction*
Anthony Fader, Stephen Soderland and Oren Etzioni

15:00–15:25 *Active Learning with Amazon Mechanical Turk*
Florian Laws, Christian Scheible and Hinrich Schütze

15:25–15:50 *Bootstrapped Named Entity Recognition for Product Attribute Extraction*
Duangmanee Putthividhya and Junling Hu

Friday, July 29, 2011 (continued)

Session 10C: Text Mining and NLP Applications
Chair: Alexandre Klementiev

14:10–14:35 *Twitter Catches The Flu: Detecting Influenza Epidemics using Twitter*
Eiji ARAMAKI, Sachiko MASKAWA and Mizuki MORITA

14:35–15:00 *A Simple Word Trigger Method for Social Tag Suggestion*
Zhiyuan Liu, Xinxiong Chen and Maosong Sun

15:00–15:25 *Rumor has it: Identifying Misinformation in Microblogs*
Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev and Qiaozhu Mei

15:25–15:50 *Exploiting Parse Structures for Native Language Identification*
Sze-Meng Jojo Wong and Mark Dras

15:50–16:20 Coffee break

Plenary session – Best Paper Award + Closing
Chair: Mark Johnson

16:20–17:10 *A Probabilistic Forest-to-String Model for Language Generation from Typed Lambda Calculus Expressions*
Wei Lu and Hwee Tou Ng

Fast and Robust Joint Models for Biomedical Event Extraction

Sebastian Riedel **Andrew McCallum**
Department of Computer Science
University of Massachusetts, Amherst
{riedel,mccallum}@cs.umass.edu

Abstract

Extracting biomedical events from literature has attracted much recent attention. The best-performing systems so far have been pipelines of simple subtask-specific local classifiers. A natural drawback of such approaches are cascading errors introduced in early stages of the pipeline. We present three joint models of increasing complexity designed to overcome this problem. The first model performs joint trigger and argument extraction, and lends itself to a simple, efficient and exact inference algorithm. The second model captures correlations between events, while the third model ensures consistency between arguments of the same event. Inference in these models is kept tractable through dual decomposition. The first two models outperform the previous best joint approaches and are very competitive with respect to the current state-of-the-art. The third model yields the best results reported so far on the BioNLP 2009 shared task, the BioNLP 2011 Genia task and the BioNLP 2011 Infectious Diseases task.

1 Introduction

Whenever we advance our scientific understanding of the world, we seek to publish our findings. The result is a vast and ever-expanding body of natural language text that is becoming increasingly difficult to leverage. This is particularly true in the context of life sciences, where large quantities of biomedical articles are published on a daily basis. To support tasks such data mining, search and visualization, there is a clear need for structured representations of the knowledge these articles convey. This is indicated by a large number of public databases with content ranging from simple protein-protein interactions to complex pathways. To increase coverage of such databases, and to keep up with the rate of publishing, we need to *automatically* extract structured representations from biomedical text—a process often referred to as biomedical text mining.

One major focus of biomedical text mining has been the extraction of named entities, such genes or gene products, and of flat binary relations between such entities, such as protein-protein interactions. However, in recent years there has also been an increasing interest in the extraction of biomedical *events* and their causal relations. This gave rise to the BioNLP 2009 and 2011 shared tasks which challenged participants to gather such events from biomedical text (Kim et al., 2009; Kim et al., 2011). Notably, these events can be complex and recursive: they may have several arguments, and some of the arguments may be events themselves.

Current state-of-the-art event extractors follow the same architectural blueprint and divide the extraction process into a pipeline of three stages (Björne et al., 2009; Miwa et al., 2010c). First they predict a set of candidate event trigger words (say, tokens 2, 5 and 6 in figure 1), then argument mentions are attached to these triggers (say, token 4 for trigger 2). The final stage decides how arguments are shared between events—compare how one event subsumes all arguments of trigger 6 in figure 1, while two events share the three arguments of trigger 4 in figure 2. This architecture is prone to cascading errors: If we miss a trigger in the first stage, we will never be able to extract the full event

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1–12,
Edinburgh, Scotland, UK, July 27–31, 2011.

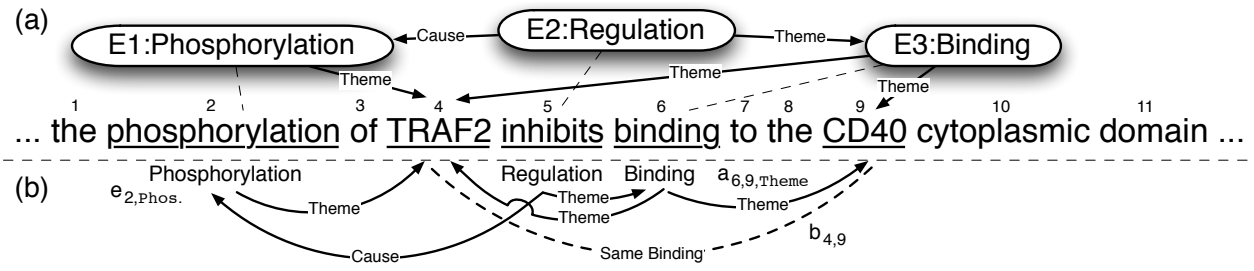


Figure 1: (a) sentence with target event structure to extract; (b) projection to a set of labelled graph over tokens.

it concerns. Current systems attempt to tackle this problem by passing several candidates to the next stage. However, this tends to increase the false positive rate. In fact, Miwa et al. (2010c) observe that 30% of their errors stem from this type of ad-hoc module communication.

Joint models have been proposed to overcome this problem (Poon and Vanderwende, 2010; Riedel et al., 2009). However, besides not being as accurate as their pipelined competitors, mostly because they do not yet exploit the rich set of features used by Miwa et al. (2010b) and Björne et al. (2009), they also suffer from the complexity of inference. For example, to remain tractable, the best joint system so far (Poon and Vanderwende, 2010) works with a simplified representation of the problem in which certain features are harder to capture, employs local search without certificates of optimality, and furthermore requires a 32-core cluster for quick train-test cycles. Existing joint models also rely on heuristics when it comes to deciding which arguments share the same event. Contrast this with the best current pipeline (Miwa et al., 2010c; Miwa et al., 2010b) which uses a classifier for this task.

We present a family of event extraction models that address the aforementioned problems. The first model jointly predicts triggers and arguments. Notably, the highest scoring event structure under this model can be found efficiently in $O(mn)$ time where m is the number of trigger candidates, and n the number of argument candidates. This is only slightly slower than the $O(m'n)$ runtime of a pipeline, where m' is the number of trigger candidates as filtered by the first stage. We achieve these guarantees through a novel algorithm that jointly picks best trigger label and arguments on a per-token basis. Remarkably, it takes roughly as much time to train this model on one core as the model of Poon and Vanderwende (2010) on 32 cores, and leads to better results.

The second model enforces additional constraints that ensure consistency between events in hierarchical regulation structures. While inference in this model is more complicated, we show how dual decomposition (Komodakis et al., 2007; Rush et al., 2010) can be used to efficiently find exact solutions for a large fraction of problems.

Our third model includes the first two, and explicitly captures which arguments are part in the same event—the third stage of existing pipelines. Due to a complex coupling between this model and the first two, inference here requires a *projected* version of the sub-gradient technique demonstrated by Rush et al. (2010).

When evaluated on the BioNLP 2009 shared task, the first two models outperform the previous best joint approaches and are competitive when compared to current state-of-the-art. With 57.4 F1 on the test set, the third model yields the best results reported so far with a 1.1 F1 margin to the results of Miwa et al. (2010b). For the BioNLP 2011 Genia task 1 and the BioNLP 2011 Infectious Diseases task, Model 3 yields the second-best and best results reported so far. The second-best results are achieved with Model 3 as is (Riedel and McCallum, 2011), the best results when using Stanford event predictions as input features (Riedel et al., 2011). The margins between Model 3 and the best runner-ups range from 1.9 F1 to 2.8 F1.

In the following we will first introduce biomedical event extraction and our notation. Then we go on to present our models and their inference routines. We present related work, show our empirical evaluation, and conclude.

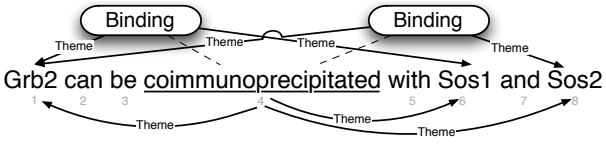


Figure 2: Two binding events with identical trigger. The projection graph does not change even if both events are merged.

2 Biomedical Event Extraction

By bio-molecular event we mean a change of state of one or more bio-molecules. Our task is to extract structured information about such events from natural language text. More concretely, let us consider part (a) of figure 1. We see a snippet of text from a biomedical abstract, and the three events that can be extracted from it. We will use these to characterize the types of events we ought to extract, as defined by the 2009 BioNLP shared task. Note that for the shared task, protein mentions are given by the task organizers and hence do not need to be extracted.

The event E1 in the figure refers to a *Phosphorylation* of the TRAF2 protein. It is an instance of a set of *simple events* that describe changes to a single gene or gene product. Other members of this set are: *Expression*, *Transcription*, *Localization*, and *Catabolism*. Each of these events has to have exactly one *theme*, the protein of which a state change is described. A labelled edge in figure 1a) shows that TRAF2 is the theme of E1.

Event E3 is a *Binding* of TRAF2 and CD40. Binding events are particular in that they may have more than one theme, as there can be several biomolecules associated in a binding structure. This is in fact the case for E3.

In the top-center of figure 1a) we see the *Regulation* event E2. Such events describe regulatory or causal relations between events. Other instances of this type of events are: *Positive Regulation* and *Negative Regulation*. Regulations have to have exactly one theme; this theme can a be protein or, as in our case, another event. Regulations may also have zero or one *cause* arguments that denote events or proteins which trigger the regulation.

In the BioNLP shared task, we are also asked to find a *trigger* (or *clue*) token for each event. This token grounds the event in text and allows users to quickly validate extracted events. For example, the trigger for event E2 is "inhibit", as indicated by a dashed line.

2.1 Event Projection

To formulate the search for event structures of the form shown in figure 1a) as an optimization problem, it will be convenient to represent them through a set of binary variables. We introduce such a representation, inspired by previous work (Riedel et al., 2009; Björne et al., 2009) and based on a projection of events to a graph structure over tokens, as seen figure 1b).

Consider sentence $\mathbf{x}$ and a set of *candidate trigger* tokens, denoted by $\text{Trig}(\mathbf{x})$. We label each candidate i with the event type it is a trigger for, or `None` if it is not a trigger. This decision is represented through a set of binary variables $e_{i,t}$, one for each possible event type t. In our example we have $e_{6,\texttt{Binding}} = 1$. The set of possible event types will be denoted as $\mathcal{T}$, the regulation event types as $\mathcal{T}_{\text{Reg}} \stackrel{\text{def}}{=} \{\texttt{PosReg}, \texttt{NegReg}, \texttt{Reg}\}$ and its complement as $\mathcal{T}_{\neg\text{reg}} \stackrel{\text{def}}{=} \mathcal{T} \setminus \mathcal{T}_{\text{Reg}}$.

For each candidate trigger i we consider the arguments of all events that have i as trigger. Each argument a will either be an event itself, or a protein. For events we add a labelled edge between i and the trigger j of a. For proteins we add an edge between i and the syntactic head j of the protein mention. In both cases we label the edge $i \rightarrow j$ with the role of the argument a. The edge is represented through a binary variable $a_{i,j,r}$, where $r \in \mathcal{R}$ is the argument role and $\mathcal{R} \stackrel{\text{def}}{=} \{\texttt{Theme}, \texttt{Cause}, \texttt{None}\}$. The role `None` is active whenever no `Theme` or `Cause` role is present. In our example we get, among others, $a_{2,4,\texttt{Theme}} = 1$.

So far our representation is equivalent to mappings in previous work (Riedel et al., 2009; Björne et al., 2009) and hence shares their main shortcoming: we cannot differentiate between two (or more) binding events with the same trigger but different arguments, or one binding event with several arguments. Consider, for example, the arguments of trigger 6 in figure 1b) that are all subsumed in a single event. By contrast, the arguments of trigger 4 shown in figure 2 are split between two events.

Previous work has resolved this ambiguity

through ad-hoc rules (Björne et al., 2009) or with a post-processing classifier (Miwa et al., 2010c). We propose to augment the graph representation through edges between pairs of proteins that are themes in the same binding event. For two protein tokens p and q we represent this edge through the binary variable $b_{p,q}$. Hence, in figure 1b) we have $b_{4,9} = 1$, whereas for figure 2 we get $b_{1,6} = b_{1,8} = 1$ but $b_{6,8} = 0$. By explicitly modeling such "sibling" edges we not only minimize the need for post-processing. We can also improve attachment decisions akin to second order models in dependency parsing (McDonald and Pereira, 2006). Note that while merely introducing such variables is easy, enforcing consistency between them and the $e_{i,t}$ and $a_{i,j,r}$ variables is not. We address this in section 3.3.1.

Reconstruction of events from solutions $(\mathbf{e}, \mathbf{a}, \mathbf{b})$ can be done almost exactly as described by Björne et al. (2009). However, while they group binding arguments according to ad-hoc rules based on dependency paths from trigger to argument, we simply query the variables $b_{p,q}$.

To simplify our exposition we introduce additional notation. We denote the set of protein head tokens with $\text{Prot}(\mathbf{x})$; the set of a possible targets for outgoing edges from a trigger is $\text{Cand}(\mathbf{x}) \stackrel{\text{def}}{=} \text{Trig}(\mathbf{x}) \cup \text{Prot}(\mathbf{x})$. We will often omit the domains of indices and instead assign them a fixed domain in advance: $i, l \in \text{Trig}(\mathbf{x})$, $j, k \in \text{Cand}(\mathbf{x})$, $p, q \in \text{Prot}(\mathbf{x})$, $r \in \mathcal{R}$ and $t \in \mathcal{T}$. Bold face letters are used to denote composite vectors $\mathbf{e}$, $\mathbf{a}$ and $\mathbf{b}$ of variables $e_{i,t}$, $a_{i,j,r}$ and $b_{p,q}$. The vector $\mathbf{y}$ is the joint vector of $\mathbf{e}, \mathbf{a}$ and $\mathbf{b}$. The short-form $\mathbf{e}_i \leftarrow t$ will mean $\forall t' : e_{i,t'} \leftarrow \delta_{t,t'}$ where $\delta_{t,t'}$ is the Kronecker Delta. Likewise, $\mathbf{a}_{i,j} \leftarrow r$ means $\forall r' : a_{i,j,r'} \leftarrow \delta_{r,r'}$.

3 Models

In this section we will present three structured prediction models of increasing complexity and expressiveness, as well as their corresponding MAP inference algorithms. Each model m can be represented by a mapping from sentence $\mathbf{x}$ to a set of *legal* structures $\mathcal{Y}_m(\mathbf{x})$, and a linear *scoring function*

$$s_m(\mathbf{y}; \mathbf{x}, \mathbf{w}) = \langle \mathbf{w}, \mathbf{f}(\mathbf{y}, \mathbf{x}) \rangle . \quad (1)$$

Here $\mathbf{f}$ is a feature function on structures $\mathbf{y}$ and input $\mathbf{x}$, and $\mathbf{w}$ is a weight vector for these features.

We can use the scoring function s_m and the set of legal structures $\mathcal{Y}_m(\mathbf{x})$ to predict the event $\mathbf{h}_m(\mathbf{x})$ for a given sentence $\mathbf{x}$ according to

$$\mathbf{h}_m(\mathbf{x}) \stackrel{\text{def}}{=} \arg \max_{\mathbf{y} \in \mathcal{Y}_m(\mathbf{x})} s_m(\mathbf{y}; \mathbf{x}, \mathbf{w}) . \quad (2)$$

For brevity we will from now on omit observations $\mathbf{x}$ and weights $\mathbf{w}$ when they are clear from the context.

3.1 Model 1

Model 1 performs a simple version of joint trigger and argument extraction. It independently scores trigger labels and argument roles:

$$s_1(\mathbf{e}, \mathbf{a}) \stackrel{\text{def}}{=} \sum_{e_{i,t}=1} s_{\text{T}}(i, t) + \sum_{a_{i,j,r}=1} s_{\text{R}}(i, j, r) . \quad (3)$$

Here $s_{\text{T}}(i, t) = \langle \mathbf{w}_{\text{T}}, \mathbf{f}_{\text{T}}(i, t) \rangle$ is a per-trigger scoring function that measures how well the event label t fits to token i. Likewise, $s_{\text{R}}(i, j, r) = \langle \mathbf{w}_{\text{R}}, \mathbf{f}_{\text{R}}(i, j, r) \rangle$ measures the compatibility of role r as label for the edge $i \rightarrow j$.

The jointness of Model 1 stems from enforcing consistency between the trigger label of i and its *outgoing* edges. By consistency we mean that: (a) there is at least one `Theme` whenever there is an event at i; (b) only regulation events are allowed to have `Cause` arguments; (c) all arguments of a `None` trigger must have the `None` role. We will denote the set assignments that fulfill these constraints by O and hence have $\mathcal{Y}_1 \stackrel{\text{def}}{=} \text{O}$.

Enforcing $(\mathbf{e}, \mathbf{a}) \in \text{O}$ guarantees that we never predict triggers i for which no sensible, high-scoring, argument j can be found. It also ensures that when we see an "obvious" argument edge $i \stackrel{r}{\rightarrow} j$ with high score $s_{\text{R}}(i, j, r)$ there is pressure to extract a trigger at i, even if the fact that i is a trigger may not be as obvious.

3.1.1 Inference

As it turns out, the maximizer of equation 2 can be found very efficiently in $O(mn)$ time where $m = |\text{Trig}(\mathbf{x})|$ and $n = |\text{Cand}(\mathbf{x})|$. The corresponding procedure, bestOut$(\cdot)$, is shown in algorithm 1. It takes as input a vector of trigger and edge *penalties* $\mathbf{c}$ that are added to the local scores of the s_{T} and s_{R} functions. For Model 2 and 3 we will use these

penalties to enforce agreement with predictions of other inference subroutines. When using Model 1 by itself we set them to $\mathbf{0}$. We point out that the scoring function s_1 is multiplied with $\frac{1}{2}$ throughout the algorithm. For doing inference in Model 1 and 2 this has no effect, but when we use bestOut$(\cdot)$ for Model 3 inference, it is required.

The bestOut $(\mathbf{c})$ routine exploits the fact that the constraints of Model 1 only act on the label for trigger i and its outgoing edges. In particular, enforcing consistency between $e_{i,t}$ and outgoing edges $a_{i,j,r}$ has no effect on consistency between $e_{l,t}$ and $a_{i',j',r'}$ for any other trigger $i' \neq i$. Moreover, for a given trigger the constraints only differentiate between three cases: (a) regulation event, (b) non-regulation event and (c) no event. This means that we can extract events on a per-trigger basis, and find the best per-trigger structure by comparing cases (a), (b) and (c). Note that bestOut $(\mathbf{c})$ uses the shorthand emptyOut (i) to denote the partial assignment $\mathbf{e}_i \leftarrow \texttt{None}$ and $\forall j : \mathbf{a}_{i,j} \leftarrow \texttt{None}$. The function $s_1^{\mathbf{c}}(i, \mathbf{y}) \stackrel{\text{def}}{=} \sum_t e_{i,t} \left(c_{i,t} + \frac{1}{2} s_{\text{T}}(i,t)\right) + \sum_{j,r} a_{i,j,r} \left(c_{i,j,r} + \frac{1}{2} s_{\text{R}}(i,j,r)\right)$ is a per-trigger frame score with penalties $\mathbf{c}$.

3.2 Model 2

Model 1 may still predict structures that cannot be mapped to events. For example, in figure 1b) we may label token 5 as `Regulation`, add the edge $5 \stackrel{\texttt{Cause}}{\rightarrow} 2$ but fail to label token 2 as an event. While consistent with $(\mathbf{e}, \mathbf{a}) \in \text{O}$, this violates the constraint that every active edge must either end at a protein, or at an active event trigger. This is a requirement on the label of a trigger and the assignment of roles for its *incoming* edges.

Model 2 enforces the above constraint in addition to $(\mathbf{e}, \mathbf{a}) \in \text{O}$, while inheriting the scoring function from Model 1. Hence, using I to denote the set of assignments with consistent trigger labels and incoming edges, we get $\mathcal{Y}_2 \stackrel{\text{def}}{=} \mathcal{Y}_1 \cap \text{I}$ and $s_2(\mathbf{y}) \stackrel{\text{def}}{=} s_1(\mathbf{y})$.

3.2.1 Inference

Inference in Model 2 amounts to optimizing $s_2(\mathbf{e}, \mathbf{a})$ over $\text{O} \cap \text{I}$. This is more involved, as we now have to ensure that when predicting an outgoing edge from trigger i to trigger l there is a high-scoring event at l. We follow Rush et al. (2010) and solve this problem in the framework of dual decomposi-

Algorithm 1 Sub-procedures for inference in Model 1, 2 and 3.

best label and outgoing edges for all triggers under penalties $\mathbf{c}$

bestOut $(\mathbf{c})$:
  $\forall i$ $\mathbf{y}^0 \leftarrow$ emptyOut (i)
    $\mathbf{y}^1 \leftarrow$ out $\left(i, \mathbf{c}, \mathcal{T}_{\text{reg}}, \mathcal{R}\right)$
    $\mathbf{y}^2 \leftarrow$ out $\left(i, \mathbf{c}, \mathcal{T}_{\neg\text{reg}}, \mathcal{R} \setminus \{\texttt{Cause}\}\right)$
    $\mathbf{y}_i \leftarrow \arg\max_{\mathbf{y} \in \{\mathbf{y}^0, \mathbf{y}^1, \mathbf{y}^2\}} s_1^{\mathbf{c}}(i, \mathbf{y})$
  return $(\mathbf{y}_i)_i$

best label and incoming edges for all triggers under penalties $\mathbf{c}$

bestIn $(\mathbf{c})$:
  $\forall l$ $\mathbf{y}^0 \leftarrow$ emptyIn (l)
    $\mathbf{y}^1 \leftarrow$ in $(l, \mathbf{c}, \mathcal{T}, \mathcal{R} \setminus \{\texttt{None}\})$
    $\mathbf{y}_l \leftarrow \arg\max_{\mathbf{y} \in \{\mathbf{y}^0, \mathbf{y}^1\}} s_2^{\mathbf{c}}(l, \mathbf{y})$
  return $(\mathbf{y}_l)_l$

pick best binding pairs p, q *and trigger* i *for each using penalties* $\mathbf{c}$

bestBind $(\mathbf{c})$:
  $\forall p, q$ $b_{p,q} \leftarrow [s_{\text{B}}(p,q) + \max_i c_{i,p,q} > 0]$
    $I_{p,q} \leftarrow \left\{i | c_{i,p,q} = \max_{i'} c_{i',p,q}\right\}$
    if $b_{p,q} = 1$ **or** $\max_{i'} c_{i',p,q} > 0$
      $\forall i : t_{i,p,q} \leftarrow [i \in I_{p,q}] \, |I_{p,q}|^{-1}$
    else
      $\forall i : t_{i,p,q} \leftarrow 0$
  return $(\mathbf{b}, \mathbf{t})$

best label in T *and outgoing edge roles in* R *for* i*, using penalties* $\mathbf{c}$

out $(i, \mathbf{c}, T, R)$:
  $\mathbf{e}_i \leftarrow \arg\max_{t \in T} \frac{1}{2} s_{\text{T}}(i,t) + c_{i,t}$
  $\mathbf{a}_{i,\text{bestTheme}(i,\mathbf{c})} \leftarrow \texttt{Theme}$
  $\forall j$ $\mathbf{a}_{i,j} \leftarrow \arg\max_{r \in R} \frac{1}{2} s_{\text{R}}(i,j,r) + c_{i,j,r}$
  return $(\mathbf{e}_i, \mathbf{a}_i)$

best label in T*, incoming edge roles in* R
and outgoing protein roles, using costs $\mathbf{c}$

in $(l, \mathbf{c}, T, R)$:
  $\mathbf{e}_l \leftarrow \arg\max_{t \in T} \frac{1}{2} s_{\text{T}}(l,t) + c_{l,t}$
  $\forall i$ $\mathbf{a}_{i,l} \leftarrow \arg\max_{r \in R} \frac{1}{2} s_{\text{R}}(i,l,r) + c_{i,l,r}$
  $\forall p$ $\mathbf{a}_{l,p} \leftarrow \arg\max_{r \in R} \frac{1}{2} s_{\text{R}}(l,p,r) + c_{l,p,r}$
  return $(\mathbf{e}_i, \mathbf{a}_i)$

best `Theme` *argument for* i

bestTheme $(i, \mathbf{c})$:
  $\text{s}(j) \stackrel{\text{def}}{=} \max_{j,r} \frac{1}{2} s_{\text{R}}(i,j,r) + c_{i,j,r}$
  $\Delta(j) \stackrel{\text{def}}{=} \frac{1}{2} s_{\text{R}}(i,j,\texttt{Theme}) + c_{i,j,\texttt{Theme}} - \text{s}(j)$
  return $\arg\max_j \Delta(j)$

tion. To this end we write our optimization problem as

$$\begin{aligned} \underset{\mathbf{e},\mathbf{a},\bar{\mathbf{e}},\bar{\mathbf{a}}}{\text{maximize}} \quad & \frac{1}{2} s_2(\mathbf{e},\mathbf{a}) + \frac{1}{2} s_2(\bar{\mathbf{e}},\bar{\mathbf{a}}) \\ \text{subject to} \quad & (\mathbf{e},\mathbf{a}) \in \mathrm{O} \wedge (\bar{\mathbf{e}},\bar{\mathbf{a}}) \in \mathrm{I} \wedge \\ & \mathbf{e} = \bar{\mathbf{e}} \wedge \mathbf{a} = \bar{\mathbf{a}} \end{aligned} \tag{M2}$$

and note that this problem could be solved separately for $\mathbf{e}, \mathbf{a}$ and $\bar{\mathbf{e}}, \bar{\mathbf{a}}$ if the coupling constraints $\mathbf{e} = \bar{\mathbf{e}}$ and $\mathbf{a} = \bar{\mathbf{a}}$ were removed.

M2 is an Integer Linear Program, as variables are binary and both objective and constraints can be represented through linear constraints.[1] Dual decomposition solves a Linear Programming (LP) relaxation of M2 (that allows fractional values for all binary variables) through subgradient descent on a particular dual of M2. This dual can be derived by introducing Lagrange multipliers for the coupling constraints. Its attractiveness stems from the fact that calculating the subgradient amounts to solving the decoupled problems in isolation. If, by design, these decoupled problems can be solved efficiently, we can often quickly find the optimal solution to an LP relaxation of our original problem.

Dual decomposition applied to Model 2 is shown in algorithm 2. It maintains the dual variables $\boldsymbol{\lambda}$ that will appear as local penalties in the subproblems to be solved. The algorithm will try to tune these variables such that at convergence the coupling constraints will be fulfilled. This is done by first optimizing $s_2(\mathbf{e},\mathbf{a})$ over O and $s_2(\bar{\mathbf{e}},\bar{\mathbf{a}})$ over I. Now, whenever there is disagreement between two variables to be coupled, the corresponding dual parameter is shifted, increasing the chance that next time both models will agree. For example, if in the first iteration we predict $e_{6,\texttt{Bind}} = 1$ but $\bar{e}_{6,\texttt{Bind}} = 0$, we set $\lambda_{6,\texttt{Bind}} = -\alpha$ where α is some stepsize (chosen according to Koo et al. (2010)). This will decrease the coefficient for $e_{6,\texttt{Bind}}$, and increase the coefficient for $\bar{e}_{6,\texttt{Bind}}$. Hence, we have a higher chance of agreement for this variable in the next iteration.

The algorithm repeats the process described above until all variables agree, or some predefined number R of iterations is reached. In the former case we in fact have the exact solution to the original ILP.

[1]The ILP representation could be taken from the MLNs of Riedel et al. (2009) and the mapping to ILPs of Riedel (2008).

Algorithm 2 Subgradient descent for Model 2, and projected subgradient descent for Model 3.

require:
R: *max. iteration,* α_t: *stepsizes*
$t \leftarrow 0$ *[model 2,3]* $\boldsymbol{\lambda} \leftarrow 0$ *[model 2,3]* $\boldsymbol{\mu} \leftarrow 0$ *[model 3]*
repeat

model	
2	$(\mathbf{e},\mathbf{a}) \leftarrow \text{bestOut}(\boldsymbol{\lambda})$
2,3	$(\bar{\mathbf{e}},\bar{\mathbf{a}}) \leftarrow \text{bestIn}(-\boldsymbol{\lambda})$
3	$(\mathbf{e},\mathbf{a}) \leftarrow \text{bestOut}(\mathbf{c}^{\text{out}}(\boldsymbol{\lambda},\boldsymbol{\mu}))$
3	$(\mathbf{b},\mathbf{t}) \leftarrow \text{bestBind}\left(\mathbf{c}^{\text{bind}}(\boldsymbol{\mu})\right)$
2,3	$\lambda_{i,t} \leftarrow \lambda_{i,t} - \alpha_t (e_{i,t} - \bar{e}_{i,t})$
2,3	$\lambda_{i,j,r} \leftarrow \lambda_{i,j,r} - \alpha_t (a_{i,j,r} - \bar{a}_{i,j,r})$
3	$\mu^{\text{trig}}_{i,p,q} \leftarrow \left[\mu^{\text{trig}}_{i,p,q} - \alpha_t (e_{i,\texttt{Bind}} - t_{i,p,q})\right]_+$
3	$\mu^{\text{arg1}}_{i,j,k} \leftarrow \left[\mu^{\text{arg1}}_{i,p,q} - \alpha_t (a_{i,p,\texttt{Theme}} - t_{i,p,q})\right]_+$
3	$\mu^{\text{arg2}}_{i,p,q} \leftarrow \left[\mu^{\text{arg2}}_{i,p,q} - \alpha_t (a_{i,q,\texttt{Theme}} - t_{i,p,q})\right]_+$
2,3	$t \leftarrow t + 1$

until no $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ changed or $t > R$
return $(\mathbf{e},\mathbf{a})$*[model 2]* or $(\mathbf{e},\mathbf{a},\mathbf{b})$ *[model 3]*

In the later case we have no such guarantee, but find that in practice the solutions are still of high quality. Notice that we could still assess the quality of this approximation by measuring the duality gap between primal score and the final dual score.

Algorithm 2 for Model 2 requires us to optimize $s_2(\mathbf{e},\mathbf{a})$ over O and $s_2(\bar{\mathbf{e}},\bar{\mathbf{a}})$ over I. The former, with added penalties, can be done with bestOut($\mathbf{c}$). As the constraint set for I again decomposes on a per-token basis, solving the latter problem requires a very similar procedure, and again $O(mn)$ time. Algorithm 1 shows this procedure under bestIn($\mathbf{c}$). It chooses, for each trigger candidate, the best label and *incoming* set of arguments together with the best outgoing edges to proteins. Adding edges to proteins is not strictly required, but simplifies our exposition. Algorithm bestIn($\mathbf{c}$) requires a per-trigger incoming score: $s^{\mathbf{c}}_2(l,\mathbf{y}_l) \stackrel{\text{def}}{=} \sum_t e_{l,t}\left(c_{l,t} + \frac{1}{2} s_{\mathrm{T}}(l,t)\right) + \sum_{i,r} a_{i,l,r}\left(c_{i,l,r} + \frac{1}{2} s_{\mathrm{R}}(i,l,r)\right) + \sum_{p,r} a_{l,p,r}\left(c_{l,p,r} + \frac{1}{2} s_{\mathrm{R}}(l,p,r)\right)$. Finally, note that emptyIn(i) not only assigns None as trigger label of i and to all incoming edges, but also greedily picks outgoing protein edges (as done within in$(\cdot)$).

3.3 Model 3

Model 2 does not predict the $b_{p,q}$ variables that represent protein pairs p, q in bindings. *Model 3* fixes this by (a) adding binding variables $b_{p,q}$ into the objective, and (b) enforcing that the binding assignment $\mathbf{b}$ is consistent with the trigger and argument assignments $\mathbf{e}$ and $\mathbf{a}$. We will also enforce that the same pair of entities p, q cannot be arguments in more than one event together.

The scoring function for Model 3 is simply

$$s_3(\mathbf{e}, \mathbf{a}, \mathbf{b}) \stackrel{\text{def}}{=} s_2(\mathbf{e}, \mathbf{a}, \mathbf{b}) + \sum_{b_{p,q}=1} s_{\text{B}}(p, q). \quad (4)$$

Here $s_{\text{B}}(p, q) = \langle \mathbf{w}_{\text{B}}, \mathbf{f}_{\text{B}}(p, q) \rangle$ is a per-protein-pair score based on a feature representation of the lexical and syntactic relation between both protein heads.

Our strategy will be based on enforcing consistency partly through linear constraints which we dualize, and partly within our search algorithm. To this end we first introduce a set of auxiliary binary variables $t_{i,p,q}$. When a $t_{i,p,q}$ is active, we enforce that there is a binding trigger at i with proteins p and q as `Theme` arguments. A set of linear constraints can be used for this: $e_{i,\texttt{Bind}} - t_{i,p,q} \geq 0$, $a_{i,p,\texttt{Theme}} - t_{i,p,q} \geq 0$ and $a_{i,q,\texttt{Theme}} - t_{i,p,q} \geq 0$ for all suitable i, p and q. We denote the set of assignments $(\mathbf{e}, \mathbf{a}, \mathbf{t})$ that fulfill these constraints by T.

Consistency between $\mathbf{e}$, $\mathbf{a}$ and $\mathbf{b}$ can now be enforced by making sure that $\mathbf{t}$ is consistent with $\mathbf{e}$ and $\mathbf{a}$, and that $\mathbf{b}$ is consistent with this $\mathbf{t}$. The latter means that an active $b_{p,q}$ requires a trigger i to point to p and q. Or in other words, $t_{i,p,q} = 1$ for exactly one trigger i.

With the set of consistent assignments $(\mathbf{b}, \mathbf{t})$ referred to as B, and a slight abuse of notation, this gives us $\mathcal{Y}_3 \stackrel{\text{def}}{=} \mathcal{Y}_2 \cap \text{T} \cap \text{B}$. Note that it is $(\mathbf{e}, \mathbf{a}, \mathbf{t}) \in \text{T}$ that will be enforced by dualizing constraints, and $(\mathbf{b}, \mathbf{t}) \in \text{B}$ that will be enforced within search.

3.3.1 Inference

We note that inference in Model 3 can be performed by solving the following problem:

$$\begin{aligned} \underset{\mathbf{e},\mathbf{a},\bar{\mathbf{e}},\bar{\mathbf{a}},\mathbf{b},\mathbf{t}}{\text{maximize}} \quad & \frac{1}{2} s_1(\mathbf{e}, \mathbf{a}) + \frac{1}{2} s_2(\bar{\mathbf{e}}, \bar{\mathbf{a}}) + \sum_{b_{p,q}=1} s_{\text{B}}(p, q) \\ \text{subject to} \quad & (\mathbf{e}, \mathbf{a}) \in \text{O} \wedge (\bar{\mathbf{e}}, \bar{\mathbf{a}}) \in \text{I} \wedge (\mathbf{b}, \mathbf{t}) \in \text{B} \wedge \\ & \mathbf{e} = \bar{\mathbf{e}} \wedge \ \mathbf{a} = \bar{\mathbf{a}} \wedge (\mathbf{e}, \mathbf{a}, \mathbf{t}) \in \text{T}. \end{aligned} \quad \text{(M3)}$$

Again, without the final row, M3 would be separable. We exploit this by performing dual decomposition with a dual objective that has multipliers $\boldsymbol{\lambda}$ for the coupling constraints and multipliers $\boldsymbol{\mu}$ for the constraints which enforce $(\mathbf{e}, \mathbf{a}, \mathbf{t}) \in \text{T}$. The resulting subgradient descent method is also shown in algorithm 2. Notably, since the constraints for T are inequalities, we require a *projected* version of the descent algorithm which enforces $\boldsymbol{\mu} \geq 0$. This manifests itself when $\boldsymbol{\mu}$ is updated using the $[\cdot]_+$ projection.

We have already described how to find the best $\mathbf{e}, \mathbf{a}$ and $\bar{\mathbf{e}}, \bar{\mathbf{a}}$ assignments. What changes for Model 3 is the derivation of the penalties for $\mathbf{e}$ and $\mathbf{a}$ that now come from both $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. We set $\mathbf{c}^{\text{out}}_{i,t}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} \lambda_{i,t} + \delta_{t,\texttt{Bind}} \sum_{p,q} \mu^{\text{trig}}_{i,p,q}$. For $j \notin \text{Prot}(\mathbf{x})$ we set $\mathbf{c}^{\text{out}}_{i,j,r}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} \lambda_{i,j,r}$; otherwise we use $\mathbf{c}^{\text{out}}_{i,j,r}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} \lambda_{i,j,r} + \sum_p \mu^{\text{arg1}}_{i,j,p} + \sum_q \mu^{\text{arg2}}_{i,q,j}$.

For finding a $(\mathbf{b}, \mathbf{t}) \in \text{B}$ that maximizes $\sum_{b_{p,q}=1} s_{\text{B}}(p, q)$ we use bestBind$(\mathbf{c})$, as shown in algorithm 1. It groups together two proteins p, q if their score plus the penalty of the best possible trigger i exceeds 0. In this case, or if there is at least one trigger with positive penalty $c_{i,p,q} > 0$, we activate the set of triggers $I(p, q)$ with maximal score.

Note that when several triggers i maximize the score, we assign them all the same fractional value $|I(p, q)|^{-1}$. This enforces the constraint that at most one binding event can point to both p and q and also means that we are solving an LP relaxation. We could enforce integer solutions and pick arbitrary triggers at a tie, but this would lower the chances of matching against predictions of other routines.

The penalties for bestBind$(\mathbf{c})$ are derived from the dual $\boldsymbol{\mu}$ by setting $\mathbf{c}^{\text{bind}}_{i,p,q}(\boldsymbol{\mu}) = -\mu^{\text{trig}}_{i,p,q} - \mu^{\text{arg1}}_{i,p,q} - \mu^{\text{arg2}}_{i,,p,q}$.

3.4 Training

We choose prediction-based passive-aggressive (PA) online learning (Crammer and Singer, 2003) with averaging to estimate the weights $\mathbf{w}$ for each of our models. PA is an error-driven learner that shifts weights towards features of the gold solution, and away from features of the current guess, whenever the current model makes a mistake.

PA learning takes into account a user-defined loss function for which we use a weighted sum

of false positives and false negatives: $l(\mathbf{y}, \mathbf{y}') \stackrel{\text{def}}{=} \text{FP}(\mathbf{y}, \mathbf{y}') + \alpha \text{FN}(\mathbf{y}, \mathbf{y}')$. We set $\alpha = 3.8$ by optimizing on the BioNLP 2009 development set.

4 Related Work

Riedel et al. (2009) use Integer Linear Programming and cutting planes (Riedel, 2008) for inference in a model similar to Model 2. By using dual decomposition instead, we can exploit tractable substructure and achieve quadratic (Model 2) and cubic (Model 3) runtime guarantees. An advantage of ILP inference are guaranteed certificates of optimality. However, in practice we also gain certificates of optimality for a large fraction of the instances we process. Poon and Vanderwende (2010) use local search and hence provide no such certificates. Their problem formulation also makes n-gram dependency path features harder to incorporate. McClosky et al. (2011b) cast event extraction as dependency parsing task. Their model assumes that event structures are trees, an assumption that is frequently violated in practice. Finally, all previous joint approaches use heuristics to decide whether binding arguments are part of the same event, while we capture these decisions in the joint model.

We follow a long line of research in NLP that addresses search problems using (Integer) Linear Programs (Germann et al., 2001; Roth and Yih, 2004; Riedel and Clarke, 2006). However, instead of using off-the-shelf solvers, we work in the framework of dual decomposition. Here we extend the approach of Rush et al. (2010) in that in addition to equality constraints we dualize more complex coupling constraints between models. This requires us to work with a projected version of subgradient descent.

While tailored towards (biomedical) event extraction, we believe that our models can also be effective in a more general Semantic Role Labeling (SRL) context. Using variants of Model 1, we can enforce many of the SRL constraints—such as "unique agent" constraints (Punyakanok et al., 2004)—without having to call out to ILP optimizers. Meza-Ruiz and Riedel (2009) showed that inducing pressure on arguments to be attached to at least one predicate is helpful; this is a soft incoming edge constraint. Finally, Model 3 can be used to efficiently capture compatibilities between semantic arguments; such compatibilities have also been shown to be helpful in SRL (Toutanova et al., 2005).

5 Experiments

We evaluate our models on several tracks of the 2009 and 2011 BioNLP shared tasks, using the official "Approximate Span Matching/Approximate Recursive Matching" F1 metric for each. We also investigate the runtime behavior of our algorithms.

5.1 Preprocessing

Each document is first processed by the Stanford CoreNLP[2] tokenizer and sentence splitter. Parse trees come from the Charniak-Johnson parser (Charniak and Johnson, 2005) with a self-trained biomedical parsing model (McClosky and Charniak, 2008), and are converted to dependency structures again using Stanford CoreNLP. Based on trigger words collected from the training set, a set of candidate trigger tokens $\text{Trig}(\mathbf{x})$ is generated for each sentence $\mathbf{x}$.

5.2 Features

The feature function $\mathbf{f}_{\text{T}}(i, t)$ extracts a per-trigger feature vector for trigger i and type $t \in \mathcal{T}$. It creates one active feature for each element in $\{t, t \in \mathcal{T}_{\text{Reg}}\} \times \text{feats}(i)$. Here $\text{feats}(i)$ denotes a collection of representations for the token i: wordform, lemma, POS tag, syntactic heads, syntactic children, and membership in two dictionaries taken from Riedel et al. (2009).

For $\mathbf{f}_{\text{R}}(i, j, r)$ we create active features for each element of $\{r\} \times \text{feats}(i, j)$. Here $\text{feats}(i, j)$ is a collection of representations of the token pair (i, j) taken from Miwa et al. (2010c) and contains: labelled and unlabeled n-gram dependency paths; edge and vertex walk features, argument and trigger modifiers and heads, words in between.

For $\mathbf{f}_{\text{B}}(p, q)$ we re-use the token pair representations from $\mathbf{f}_{\text{R}}$. In particular, we create one active feature for each element in $\text{feats}(p, q)$.

5.3 Shared Task 2009

We first evaluate our models on the Bionlp 2009 task 1. The training, development and test sets for this

[2] http://nlp.stanford.edu/software/corenlp.shtml

	SVT	BIND	REG	TOT
McClosky	75.4	**48.4**	40.4	53.5
Poon	77.5	47.9	44.1	55.5
Bjoerne	77.9	42.2	45.5	55.7
Miwa	78.6	46.9	47.7	57.8
M1	77.2	43.0	45.8	56.2
M2	77.9	42.4	47.6	57.2
M3	**78.4**	48.0	**49.1**	**58.7**

Table 1: F1 scores for the development set of Task 1 of the BioNLP 2009 shared task.

	SVT	BIND	REG	TOT
McClosky	68.3	46.9	33.3	48.6
Poon	69.5	42.5	37.5	50.0
Bjoerne	70.2	44.4	40.1	52.0
Miwa	72.1	50.6	45.3	56.3
M1	71.0	42.1	41.9	53.4
M2	70.5	41.3	43.6	53.7
M3	71.1	**52.9**	45.2	55.8
M3+enju	**72.6**	52.6	**46.9**	**57.4**

Table 2: F1 scores for the test set of Task 1 of the BioNLP 2009 shared task.

task consist of 797, 150 and 250 documents, respectively.

Table 1 shows our results for the development set. We compare our three models (M1, M2 and M3) and previous state-of-the-art systems: *McClosky* (McClosky et al., 2011a), *Poon* (Poon and Vanderwende, 2010), *Bjoerne* (Björne et al., 2009) and *Miwa* (Miwa et al., 2010b; Miwa et al., 2010a). Presented is F1 score for all events (TOT), regulation events (REG), binding events (BIND) and simple events (SVT).

Model 1 is outperforming the previous best joint models of Poon and Vanderwende (2010), as well as the best entry of the 2009 task (Björne et al., 2009). This is achieved without careful tuning of thresholds that control flow of information between trigger and argument extraction. Notably, training Model 1 takes approximately 20 minutes using a single core implementation. Contrast this with 20 minutes on 32 cores reported by Poon and Vanderwende (2010).

Model 2 focuses on regulation structures and results demonstrate this: F1 for regulations goes up by nearly 2 points. While the impact of joint modeling relative to weaker local baselines has been shown shown by Poon and Vanderwende (2010) and Riedel et al. (2009), our findings here provide evidence that it remains effective even when the baseline system is very competitive.

With Model 3 our focus is extended to binding events, improving F1 for such events by at least 5 F1. This also has a positive effect on regulation events, as regulations of binding events can now be more accurately extracted. In total we see a 1.1 F1 increase over the best results reported so far (Miwa et al., 2010b). Crucially, this is achieved using only a single parse tree per sentence, as opposed to three used by Miwa et al. (2010a).

Table 2 shows results for the test set. Here with Model 1 we again already outperform all but the results of Miwa et al. (2010a). Model 2 improves F1 for regulations, while Model 3 again increases F1 for both regulations and binding events. This yields the best binding event results reported so far. Notably, not only are we able to resolve binding ambiguity better. Binding attachments themselves also improve, as we increase attachment F1 from 61.4 to 62.7 when going from Model 2 to Model 3.

Miwa et al. (2010b) use two parsers to generate their input features. For fairer comparison we augment Model 3 with syntactic features based on the *enju* parser (Miyao et al., 2009). With these features (M3+enju) we achieve the best results on this dataset reported so far, and outperform Miwa et al. (2010b) by 1.1 F1 in total, 1.6 F1 on regulation events and 2.0 F1 on binding events.

We also apply Model 3, with slight modifications, to the BioNLP 2009 task 2 which requires cellular locations to be extracted as well. With 53.0 F1 we fall 2 points short of the results of Miwa et al. (2010b) but still substantially outperform any other reported results on the dataset. More parse trees may again substantially improve results, as well as task-specific constraint and feature sets.

5.4 Shared Task 2011

We entered the Shared Task 2011 with Model 3, primarily focusing on Genia track (task 1), and the Infectious Diseases track. The Genia track differs from the 2009 task by including both abstracts and full text articles. In total 908 training, 259 development and 347 test documents are provided.

Genia Task 1		Infectious Diseases	
System	TOT	System	TOT
M3+Stanford	**56.0**	M3+Stanford	**55.6**
M3	55.2	M3	53.4
UTurku	53.3	Stanford	50.6
MSR-NLP	51.5	UTurku	44.2
ConcordU	50.3	PNNL	42.6

Table 3: F1 scores for the test sets of two tracks in the BioNLP 2011 Shared Task.

The top five entries are shown in table 3. Model 3 is the best-performing system that does not use model combination, only outperformed by a version of Model 3 that includes Stanford predictions (McClosky et al., 2011b) as input features (Riedel et al., 2011). Not shown in the table are results for full papers only. Here M3 ranks first with 53.1 F1, while M3+Stanford comes in second with 52.7 F1.

The Infectious Diseases (ID) track of the 2011 task has 152 train, 46 development and 118 test documents. Relative to Genia it provides less data and introduces more types of entities as well as the *biological process* event type. Incorporating these changes into our models is straightforward, and hence we omit details for brevity.

Table 3 shows the top five entries for the Infectious Diseases track. Again Model 3 is the best-performing system that does not use model combination, outperformed only by Model 3 with Stanford predictions as features. We should point out that the feature sets and learning parameters were kept constant when moving from Genia to ID data. The strong results we observe without any tuning to the domain indicate the robustness of joint modeling.

5.5 Runtime Behavior

Table 4 shows the asymptotic complexity of our three models with respect to $m = |\text{Trig}(\mathbf{x})|$, $n = |\text{Cand}(\mathbf{x})|$ and $p = |\text{Prot}(\mathbf{x})|$. We also show the number of iterations needed on average, the average time in milliseconds per sentence,[3] and the fraction of sentences we get certificates of optimality for.

As expected, Model 1 is most efficient, both asymptotically and on average. Given that its accuracy is already good, it can serve as a basis for large-scale extraction tasks. Models 2 and 3 require several iterations and more time, while providing slightly less certificates. However, given the improvement in F1 they deliver, and the fact preprocessing steps such as parsing would still dominate the average time, this seems like a reasonable price to pay.

[3]Measured without preprocessing and feature extraction.

	Complexity	Iter.	Time	Exact
M1	$O(nm)$	1.0	60ms	100%
M2	$O(Rnm)$	10.4	183ms	96%
M3	$O\left(Rnm + Rp^2m\right)$	11.7	297ms	94%

Table 4: Complexity and Runtime Behavior.

6 Conclusion

We presented three joint models for biomedical event extraction. Model 1 reaches near-state-of-the-art results, outperforms all previous joint models and has quadratic runtime guarantees. By explicitly capturing regulation events (Model 2), and binding events (Model 3) we achieve the best results reported so far on several event extraction tasks. The runtime penalty we pay is kept minimal by using dual decomposition. We also show how dual decomposition can be used for constraints that go beyond coupling equalities.

We use joint models, a decomposition technique and supervised online learning. This recipe can be successful in many settings, but requires expensive manual annotation. In the future we want to integrate weak supervision techniques to train extractors with existing biomedical databases, such as KEGG, and only minimal amounts of annotated text.

Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval. The University of Massachusetts gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP '09)*, pages 10–18, Morristown, NJ, USA. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 173–180.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL '01)*, pages 228–235.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP '09)*.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, and Jun'ichi Tsujii. 2011. Overview of BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, Portland, Oregon, June. Association for Computational Linguistics.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. Mrf optimization via dual decomposition: Message-passing revisited. In *Proceedings of the 11st IEEE International Conference on Computer Vision (ICCV '07)*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with nonprojective head automata. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '10)*.

David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL '08)*.

David McClosky, Mihai Surdeanu, and Chris Manning. 2011a. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL '11)*, Portland, Oregon, June.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011b. Event extraction as dependency parsing in bionlp 2011. In *BioNLP 2011 Shared Task*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the ACL (EACL '06)*, pages 81–88.

Ivan Meza-Ruiz and Sebastian Riedel. 2009. Jointly identifying predicates, arguments and senses using markov logic. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '09)*.

Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010a. A comparative study of syntactic parsers for event extraction. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, BioNLP '10, pages 37–45, Stroudsburg, PA, USA. Association for Computational Linguistics.

Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010b. Evaluating dependency representation for event extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 779–787, Stroudsburg, PA, USA. Association for Computational Linguistics.

Makoto Miwa, Rune Saetre, Jin-Dong D. Kim, and Jun'ichi Tsujii. 2010c. Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology*, 8(1):131–146, February.

Yusuke Miyao, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics/computer Applications in The Biosciences*, 25:394–400.

Hoifung Poon and Lucy Vanderwende. 2010. Joint Inference for Knowledge Extraction from Biomedical Literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821, Los Angeles, California, June. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, Wen tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics (COLING '04)*, pages 1346–1352, Morristown, NJ, USA. Association for Computational Linguistics.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '06)*, pages 129–137.

Sebastian Riedel and Andrew McCallum. 2011. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the*

Natural Language Processing in Biomedicine NAACL 2011 Workshop (BioNLP '11), June.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2009 Workshop (BioNLP '09)*, pages 41–49.

Sebastian Riedel, David McClosky, Mihai Surdeanu, Christopher D. Manning, and Andrew McCallum. 2011. Model combination for event extraction in BioNLP 2011. In *Proceedings of the Natural Language Processing in Biomedicine NAACL 2011 Workshop (BioNLP '11)*, June.

Sebastian Riedel. 2008. Improving the accuracy and efficiency of MAP inference for markov logic. In *Proceedings of the 24th Annual Conference on Uncertainty in AI (UAI '08)*, pages 468–475.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL' 04)*, pages 1–8.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP '10)*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 589–596, Morristown, NJ, USA. Association for Computational Linguistics.

Predicting Thread Discourse Structure over Technical Web Forums

Li Wang,[♠♡] **Marco Lui,**[♠♡] **Su Nam Kim,**[♠♡] **Joakim Nivre**[♢] and **Timothy Baldwin**[♠♡]
♠ Dept. of Computer Science and Software Engineering, University of Melbourne
♡ NICTA Victoria Research Laboratory
♢ Dept. of Linguistics and Philology, Uppsala University
li.wang.d@gmail.com, saffsd@gmail.com,
sunamkim@gmail.com, joakim.nivre@lingfil.uu.se, tb@ldwin.net

Abstract

Online discussion forums are a valuable means for users to resolve specific information needs, both interactively for the participants and statically for users who search/browse over historical thread data. However, the complex structure of forum threads can make it difficult for users to extract relevant information. The discourse structure of web forum threads, in the form of labelled dependency relationships between posts, has the potential to greatly improve information access over web forum archives. In this paper, we present the task of parsing user forum threads to determine the labelled dependencies between posts. Three methods, including a dependency parsing approach, are proposed to jointly classify the links (relationships) between posts and the dialogue act (type) of each link. The proposed methods significantly surpass an informed baseline. We also experiment with "in situ" classification of evolving threads, and establish that our best methods are able to perform equivalently well over partial threads as complete threads.

1 Introduction

Web user forums (or simply "forums") are online platforms for people to discuss information and obtain information via a text-based threaded discourse, generally in a pre-determined domain (e.g. IT support or DSLR cameras). With the advent of Web 2.0, there has been an explosion of web authorship in this area, and forums are now widely used in various areas such as customer support, community development, interactive reporting and online eduction. In addition to providing the means to interactively participate in discussions or obtain/provide answers to questions, the vast volumes of data contained in forums make them a valuable resource for "support sharing", i.e. looking over records of past user interactions to potentially find an immediately applicable solution to a current problem. On the one hand, more and more answers to questions over a wide range of domains are becoming available on forums; on the other hand, it is becoming harder and harder to extract and access relevant information due to the sheer scale and diversity of the data.

This research aims at enhancing information access and support sharing, by mining the discourse structure of troubleshooting-oriented web user forum threads. Previous research has shown that simple thread structure information (e.g. reply-to structure) can enhance tasks such as forum information retrieval (Seo et al., 2009) and post quality assessment (Lui and Baldwin, 2009). We aim to move beyond simple threading, to predict not only the links between posts, but also show the manner of each link, in the form of the discourse structure of the thread. In doing so, we hope to be able to perform richer visualisation of thread structure (e.g. highlighting the key posts which appear to have led to a successful resolution to a problem), and more fine-grained weighting of posts in threads for search purposes.

To illustrate the task, we use an example thread, made up of 5 posts from 4 distinct participants, from the CNET forum dataset of Kim et al. (2010b), as shown in Figure 1. The discourse structure of the thread is modelled as a rooted directed acyclic graph

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 13–25,
Edinburgh, Scotland, UK, July 27–31, 2011.

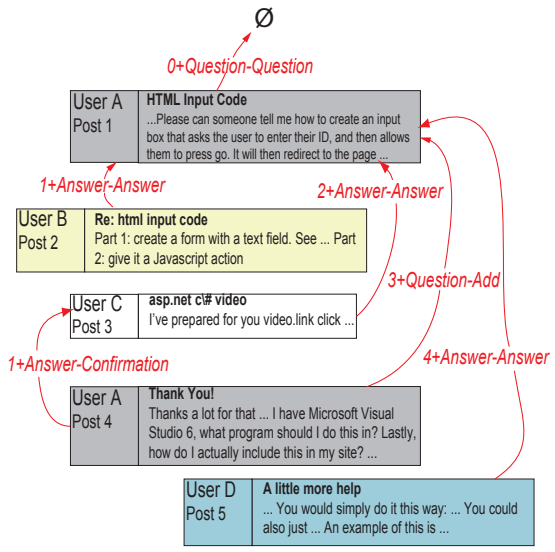


Figure 1: A snippeted and annotated CNET thread

(DAG) with a dialogue act label associated with each edge of the graph. In this example, UserA initiates the thread with a question (dialogue act = Question-Question) in the first post, by asking how to create an interactive input box on a webpage. In response, UserB and UserC provide independent answers (dialogue act = Answer-Answer). UserA responds to UserC to confirm the details of the solution (dialogue act = Answer-Confirmation), and at the same time, adds extra information to his/her original question (dialogue act = Question-Add); i.e., this one post has two distinct dependency links associated with it. Finally, UserD proposes a different solution again to the original question.

To predict thread discourse structure of this type, we jointly classify the links and dialogue acts between posts, experimenting with a variety of supervised classification methods, namely dependency parsing and linear-chain conditional random fields. In this, we build on the earlier work of Kim et al. (2010b) who first proposed the task of thread discourse analysis, but only carried out experiments on post linking and post dialogue act classification as separate tasks. In addition to achieving state-of-the-art accuracy over the task, we carry out in-depth analysis of classification effectiveness at different thread depths, and establish that the accuracy of our method over partial threads is equivalent to that over full threads, indicating that the method is applicable to in-situ thread classification. Finally, we investigate the role of user-level features in discourse structure analysis.

2 Related Work

This work builds directly on earlier work of a subset of the authors (Kim et al., 2010b), whereby a novel post-level dialogue act set was proposed, and used as the basis for annotation of a set of threads taken from CNET. In the original work, we proposed a set of novel features, which we applied to the separate tasks of post link classification and dialogue act classification. We later applied the same basic methodology to dialogue act classification over one-on-one live chat data with provided message dependencies (Kim et al., 2010a), demonstrating the generalisability of the original method. In both cases, however, we tackled only a single task, either link classification (optionally given dialogue act tags) or dialogue act classification, but never the two together. In this paper, we take the obvious step of exploring joint classification of post link and dialogue act tags, to generate full thread discourse structures.

Discourse disentanglement (i.e. link classification) and dialogue act tagging have been studied largely as independent tasks. Discourse disentanglement is the task of dividing a conversation thread (Elsner and Charniak, 2008; Lemon et al., 2002) or document thread (Wolf and Gibson, 2005) into a set of distinct sub-discourses. The disentangled discourse is sometimes assumed to take the form of a tree structure (Grosz and Sidner, 1986; Lemon et al., 2002; Seo et al., 2009), an acyclic graph structure (Rosé et al., 1995; Schuth et al., 2007; Elsner and Charniak, 2008; Wang et al., 2008; Lin et al., 2009), or a more general cyclic chain graph structure (Wolf and Gibson, 2005). Dialogue acts are used to describe the function or role of an utterance in a discourse, and have been applied to the analysis of mediums of communication including conversational speech (Stolcke et al., 2000; Shriberg et al., 2004; Murray et al., 2006), email (Cohen et al., 2004; Carvalho and Cohen, 2005; Lampert et al., 2008), instant messaging (Ivanovic, 2008; Kim et al., 2010a), edited documents (Soricut and Marcu, 2003; Sagae, 2009) and online forums (Xi et al.,

2004; Weinberger and Fischer, 2006; Wang et al., 2007; Fortuna et al., 2007; Kim et al., 2010b). For a more complete review of models for discourse disentanglement and dialogue act tagging, see Kim et al. (2010b).

Joint classification has been applied in a number of different contexts, based on the intuition that it should be possible to harness interactions between different sub-tasks to the mutual benefit of both. Warnke et al. (1997) jointly performed segmentation and dialogue act classification over a German spontaneous speech corpus. In their approach, the predictions of a multi-layer perceptron classifier on dialogue act boundaries were fed into an n-gram language model, which was used for the joint segmentation and classification of dialogue acts. Sutton and McCallum (2005) performed joint parsing and semantic role labelling (SRL), using the results of a probabilistic SRL system to improve the accuracy of a probabilistic parser. Finkel and Manning (2009) built a joint, discriminative model for parsing and named entity recognition (NER), addressing the problem of inconsistent annotations across the two tasks, and demonstrating that NER benefited considerably from the interaction with parsing. Dahlmeier et al. (2009) proposed a joint probabilistic model for word sense disambiguation (WSD) of prepositions and SRL of prepositional phrases (PPs), and achieved state-of-the-art results over both tasks.

There has been a recent growth in user-level research over forums. Lui and Baldwin (2009) explored a range of user-level features, including replies-to and co-participation graph analysis, for post quality classification. Lui and Baldwin (2010) introduced a novel user classification task where each user is classified against four attributes: clarity, proficiency, positivity and effort. User communication roles in web forums have also been studied (Chan and Hayes, 2010; Chan et al., 2010).

Threading information has been shown to enhance retrieval effectiveness for post-level retrieval (Xi et al., 2004; Seo et al., 2009), thread-level retrieval (Seo et al., 2009; Elsas and Carbonell, 2009), sentence-level shallow information extraction (Sondhi et al., 2010), and near-duplicate thread detection (Muthmann et al., 2009). These results suggest that the thread structural representation used in this research, which includes both linking structure and the dialogue act associated with each link, could potentially provide even greater leverage in these retrieval tasks.

Another related research area is post-level classification, such as general post quality classification (Weimer et al., 2007; Weimer and Gurevych, 2007; Wanas et al., 2008; Lui and Baldwin, 2009), and post descriptiveness in particular domains (e.g. medical forums: Leaman et al. (2010)). It has been demonstrated (Wanas et al., 2008; Lui and Baldwin, 2009) that thread discourse structure can significantly improve the classification accuracy for post-level tasks.

Initiation–response pairs (e.g. question–answer, assessment–agreement, and blame–denial) from online forums have the potential to enhance thread summarisation or automatically generate knowledge bases for Community Question Answering (cQA) services such as Yahoo! Answers. While initiation–response pair identification has been explored as a pairwise ranking problem (Wang and Rosé, 2010), question–answer pair identification has been approached via the two separate sub-tasks of question classification and answer detection (Cong et al., 2008; Ding et al., 2008; Cao et al., 2009). Our thread discourse structure prediction task includes joint classification of post roles (i.e. dialogue acts) and links, and could potentially be performed at the sub-post sentence level to extract initiation–response pairs.

3 Task Description and Data Set

The main task performed in this research is joint classification of inter-post links (Link) and dialogue acts (DA) within forum threads. In this, we assume that a post can only link to an earlier post (or a virtual root node), and that dialogue acts are labels on edges. It is possible for there to be multiple edges from a given post, e.g. if a post both confirms the validity of an answer and adds extra information to the original question (as happens in Post4 in Figure 1).

We experiment with two different approaches to joint classification: (1) a linear-chain CRF over combined Link/DA post labels; and (2) a dependency parser. The joint classification task is a natural fit for dependency parsing, in that the task is intrinsically one of inferring labelled dependencies

between posts, but it has a number of special properties that distinguish it from standard dependency parsing:

strict reverse-chronological directionality: the head always precedes the dependent, in terms of the chronological sequencing of posts.

non-projective dependencies: threads can contain non-projective dependencies, e.g. in a 4-post thread, posts 2 and 3 may be dependent on post 1, and post 4 dependent on post 2; around 2% of the threads in our dataset contain non-projective dependencies.

multi-headedness: it is possible for a given post to have multiple heads, including the possibility of multiple dependency links to the same post (e.g. adding extra information to a question [Question-Add] as well as retracting information from the original question [Question-Correction]); around 6% of the threads in our dataset contain multi-headed dependencies.

disconnected sub-graphs: it is possible for there to be disconnected sub-graphs, e.g. in instances where a user hijacks a thread to ask their own unrelated question, or submit an unrelated spam post; around 2% of the threads in our dataset contain disconnected sub-graphs.

The first constraint potentially simplifies dependency parsing, and non-projective dependencies are relatively well understood in the dependency parsing community (Tapanainen and Jarvinen, 1997; McDonald et al., 2005). Multi-headedness and disconnected sub-graphs pose greater challenges to dependency parsing, although there has been research done on both (McDonald and Pereira, 2006; Sagae and Tsujii, 2008; Eisner and Smith, 2005). The combination of non-projectivity, multi-headedness and disconnected sub-graphs in a single dataset, however, poses a challenge for dependency parsing.

In addition to performing evaluation in batch mode over complete threads, we consider the task of "in situ thread classification", whereby we predict the discourse structure of a thread after each post. This is intended to simulate the more realistic setting of incrementally crawling/updating thread data, but needing to predict discourse structure for partial threads. We are interested in determining the relative degradation in accuracy for in situ classification vs. batch classification.

As our dataset, we use the CNET forum dataset of Kim et al. (2010b),[1] which contains 1332 annotated posts spanning 315 threads, collected from the Operating System, Software, Hardware and Web Development sub-forums of cnet.[2] Each post is labelled with one or more links (including the possibility of null-links, where the post doesn't link to any other post), and each link is labelled with a dialogue act. The dialogue act set is made up of 5 super-categories: Question, Answer, Resolution (confirmation of the question being resolved), Reproduction (external confirmation of a proposed solution working) and Other. The Question category contains 4 sub-classes: Question, Add, Confirmation and Correction. Similarly, the Answer category contains 5 sub-classes: Answer, Add, Confirmation, Correction and Objection. For example, the label Question-Add signifies the Question superclass and Add subclass, i.e. addition of extra information to a question. For full details of the dialogue act tagset, see Kim et al. (2010b).

Dependency links are represented by their relative position in the chronologically-sorted list of posts, e.g. 1 indicates a link back to the preceding post, and 2 indicates a link back two posts.

Unless otherwise noted, evaluation is over the combined link and dialogue act tag, including the combination of superclass and subclass for the Question and Answer dialogue acts. For example, 1+Answer-Answer indicates a dependency link back one post, which is an answer to a question. The most common label in the dataset is 1+Answer-answer (28.4%).

4 Learners and Features

4.1 Learners

To predict thread discourse structure, we use a structured classification approach — based on the findings of Kim et al. (2010b) and Kim et al. (2010a) — and a dependency parser. The structured classification approach we experiment with is a linear-

[1] Available from http://www.csse.unimelb.edu.au/research/lt/resources/conll2010-thread/

[2] http://forums.cnet.com/

chain conditional random field learner (CRF: Lafferty et al. (2001)), within which we explore two simple approaches to joint classification, as is explained in Section 5.1. Dependency parsing (Kübler et al., 2009) is the task of automatically predicting the dependency structure of a token sequence, in the form of binary asymmetric dependency relations with dependency types.

Standardly, CRFs have been applied to tasks such as part-of-speech tagging, named entity recognition, semantic role labelling and supertagging, where the individual tokens are single words. Similarly, dependency parsing is conventionally applied to sentences, with single-word tokens. In our case, our tokens are thread posts, with much greater scope for feature engineering than single words, and technical challenges in scaling the underlying implementations to handle potentially much larger feature sets.

As our learners, we deployed CRFSGD (Bottou, 2011) to learn the CRF, and MaltParser (Nivre et al., 2007) as our dependency parser. CRFSGD uses stochastic gradient descent to efficiently solve the convex optimisation problem, and scales well to large feature sets. We used the default parameter settings for CRFSGD, with feature templates including all unigram features of the current token as well as bigram features combining the previous output token with the current token.

MaltParser implements transition-based parsing, where no formal grammar is considered, and a transition system, or state machine, is learned to map a sentence onto its dependency graph. One feature of MaltParser that makes it well suited to our task is that it is possible to define feature models of arbitrary complexity for each token. In presenting the thread data to MaltParser, we represent the null-link from the initial post of each thread, as well as any disconnected posts, as the root.

To the best of our knowledge, there is no past work on using dependency parsing to learn thread discourse structure. Based on extensive experimentation, we determined that the MaltParser configuration that obtains the best results for our task is the Nivre algorithm in arc-standard mode (Nivre, 2003; Nivre, 2004), using LIBSVM (Chang and Lin, 2011) with a linear kernel as the learner, and a feature model with exhaustive combinations of features relating to the features and predictions of the first/top three tokens from both "Input" and "Stack".[3] As such, MaltParser is actually unable to predict any non-projective structures, as experiments with algorithms supporting non-projective structures invariably led to lower results. In our choice of parsing algorithm, we are also unable to detect posts with multiple heads, but can potentially detect disconnected sub-graphs.

4.2 Features

The features used in our classifiers are as follows:

Structural Features:

Initiator a binary feature indicating whether the current post's author is the thread initiator.

Position the relative position of the current post, as a ratio over the total number of posts in the thread.

Semantic Features:

TitSim the relative location of the post which has the most similar title (based on unweighted cosine similarity) to the current post.

PostSim the relative location of the post which has the most similar content (based on unweighted cosine similarity) to the current post.

Punct the number of question marks (QuCount), exclamation marks (ExCount) and URLs (UrlCount) in the current post.

UserProf the class distribution (in the training thread) of the author of the current post.

These features are drawn largely from the work of Kim et al. (2010b), with two major differences: (1) we do not use post context features because our learners (i.e. CRFSGD and MaltParser) inherently capture Markov chains; and (2) our UserProf features are customised to the class set associated with the task at hand, e.g. the UserProf features for the standalone linking task take the form of the link labels (and not dialogue act labels) of the posts by the relevant author in the training data. Table 1 shows the feature representation of the third post in a thread

Feature	Value	Explanation
Initiator	1.0	post from the initiator
ExCount	4.0	4 exclamation marks
QuCount	0.0	0 question marks
UrlCount	0.0	0 URLs
Position	0.25	$\frac{i-1}{n} = \frac{3-1}{8}$
PostSim	2.0	most similar to post 1
TitSim	2.0	most similar to post 1
UserProf	$\vec{x}$	counts for posts of each class from the same author in the training data

Table 1: The feature presentation of the third post in a thread of length 8

of length 8. The values of each feature are scaled to the range $[0, 1]$ before being fed into the learners.

We also experimented with other features, including raw bag-of-words lexical features, dimensionality-reduced lexical features (using principal components analysis), and different post similarity measures such as longest common subsequence (LCS) match. While we were able to obtain gains in isolation, when combined with the other features, these features had no impact, and are thus not included in the results presented in this paper.

5 Classification Methodology

All our experiments were carried out based on stratified 10-fold cross-validation, stratifying at the thread level to ensure that all posts from a given thread occur in a single fold. The results are primarily evaluated using post-level micro-averaged F-score (F_μ: $\beta = 1$), and additionally with thread-level F-score/classification accuracy (i.e. the proportion of threads where all posts have been correctly classified[4]), where space allows. Statistical significance is tested using randomised estimation (Yeh, 2000) with $p < 0.05$. Initial experiments showed it is hard for learners to discover which posts have multiple links, largely due to the sparsity of multi-headed posts (which account for less than 5% of the total posts). Therefore, only the the most recent link for each multi-headed post was included in training, but evaluation still considers all links.

5.1 Joint classification

In our experiments, we test two basic approaches to joint classification for the CRF: (1) classifying the Link and DA separately, and composing the predictions to form the joint classification (Composition); and (2) combining the Link and DA labels into a single class, and applying the learner over the posts with the combined class (Combine). Note that Composition has the potential for mismatches in the number of Link and DA predictions it generates, causing complications in the class composition. Even if the same number of labels is predicted for both Link and DA, if multiple tags are predicted in both cases, we are left with the problem of determining which link label to combine with which dialogue act label. As such, we have our reservations about Composition, but as the CRF performs strict 1-of-n labelling, these are not issues in the experiments reported herein.

MaltParser natively handles the combination of Link and DA in its dependency parsing formulation.

5.2 In Situ Thread Classification

One of the biggest challenges in classifying the discourse structure of a forum thread is that threads evolve over time, as new posts are posted. In order to capture this phenomenon, and compare the accuracy of different models when applied to partial thread data (artificially cutting off a thread at post N) vs. complete threads.[5] This is done in the following way: classification over the first two posts only ($[1, 2]$), the first four posts ($[1, 4]$), the first six posts ($[1, 6]$), the first eight posts ($[1, 8]$), and all posts ($[all]$). In each case, we limit the test data only, meaning that the only variable in play is the extent of thread context used to learn the thread discourse structure for the given set of posts. We break down the results in each case into the indicated subthreads, e.g. we take the predictions for $[all]$, and break them down into the results for $[1, 2]$, $[1, 4]$, $[1, 6]$, $[1, 8]$ and $[all]$, for direct comparison with the predictions over the respective sub-thread data.

[3] `http://maltparser.org/userguide.html#parsingalg`

[4] Classification accuracy = F-score at the thread-level, as each thread is assigned a single label of correct or incorrect.

[5] In practice, completeness is defined at a given point in time, when the crawl was done, and it is highly likely that some of the "complete" threads had extra posts after the crawl.

Method	Link	DA
Kim et al. (2010b)	.863 / .676	.751 / .543
CRFSGD	.891 / .727	.795 / .609

Table 2: Post/thread-level component-wise classification F-scores for Link and DA classes

6 Experiments and Analysis

6.1 Joint classification

As our baseline for the task, we first use a simple majority class classifier in the form of the single joint class of 1+Answer-Answer for all posts, which has a post-level F-score of 0.284. A stronger baseline is to classify all first posts as 0+Question-Question and all subsequent posts as 1+Answer-answer, which achieves a post-level F-score of 0.515 (labelled as Heuristic).

As described in Section 5.1, one approach to joint classification with CRFSGD is to firstly conduct component-wise classification over Link and DA separately, and compose the predictions. The results for the separate Link and DA classification tasks are presented in Table 2, along with the best results for Link and DA classification from Kim et al. (2010b). At the component-wise tasks, our method is superior to Kim et al. (2010b), based on a different learner and slightly different feature set.

Next, we compose the component-wise classifications for the CRF into joint classifications (Composition). We contrast this with the combined class approach for CRFSGD and MaltParser (jointly presented as Joint in Table 3). With the combined class results, we additionally ablate each of the feature types from Section 4.2, and also present results for a dummy model, where no features are provided and the prediction is based simply on sequential priors (Dummy). The results are presented in Table 3, along with the Heuristic baseline result.

Several interesting things can be observed from the post-level F-score results in Table 3. First, with no features (Dummy), while CRFSGD performs slightly worse than the Heuristic baseline, MaltParser significantly surpasses the baseline. This is due to the richer sequential context model of MaltParser. Second, the single feature with the greatest impact on results is UserProf, i.e. user profile features extracted from the training data; CRFSGD in particular benefits from this feature. We return to explore this effect in Section 6.4. Third, although the Initiator feature does not have much effect on CRFSGD, it affects the performance of MaltParser significantly. Further experiments shown that the combination of Initiator and UserProf is sufficient to achieve a competitive result (i.e. 0.731). It therefore seems that MaltParser is more robust than CRFSGD, whose performance relies crucially on user-level features which must be learned from the training data (i.e. UserProf).

Method	CRFSGD	MaltParser
Heuristic	.515*/ .311*	
Dummy	.508*/ .394*	.533*/ .356*
Composition	.728*/ .553*	—
Joint +ALL	.756 / .578	.738 / .578
−Initiator	.745 / .569	.708*/ .534*
−Position	.750 / .565	.736 / .568
−PostSim	.753 / .578	.737 / .568
−TitSim	.760 / .587	.734 / .571
−Punct	.745 / .571	.735 / .578
−UserProf	.672*/ .527*	.701*/ .536*

Table 3: Post/thread-level Link-DA joint classification F-scores ("*" signifies a significantly worse result than that for the same learner with ALL features)

Looking to the thread-level F-scores, we observe some interesting divergences from the post-level F-score results. First, with no features (Dummy), CRFSGD significantly outperforms both the baseline and MaltParser. This appears to be because CRFSGD performs particularly well over short threads (e.g. of length 3 and 4), but worse over longer threads. Second, the best thread-level F-scores from CRFSGD (i.e. 0.587) and MaltParser (i.e. 0.578) are not significantly different, despite the discrepancy in post-level F-score (where CRFSGD is markedly superior in this case). With the extra features, the performance of MaltParser on short threads appears to pick up noticeably, and the difference in post-level predictions is over longer threads.

If we evaluate the two models over DA superclasses only (ignoring mismatches at the subclass level for Question and Answer), the post-level F-scores for joint classification with ALL features for CRFSGD and MaltParser are 0.803 and 0.787, respectively.

Approaches	Link	DA
Component-wise	.891 / .727*	**.795 / .609**
CRFSGD decomp	**.893 / .749**	.785 / .603
MaltParser decomp	.870*/ .730*	.766*/ .571*

Table 4: Post/thread-level Link and DA F-scores from component-wise classification, and from Link-DA classification decomposition ("*" signifies a significantly *worse* result than the **best** result in that column)

Looking at the performance of CRFSGD (in Combine mode) and MaltParser on disconnected sub-graphs, while both models did predict a small number of non-initial posts with null-links (including MaltParser predicting 5 out of 6 posts in a single thread as having null-links), none were correct, and neither model was able to correctly predict any of the 6 actual non-initial instances of null-links in the dataset.

Finally, we took the joint classification results from CRFSGD and MaltParser using ALL features, and decomposed the predictions into Link and DA. The results are presented in Table 4, along with the results for component-wise classification from Table 2. Somewhat surprisingly, the decomposed predictions are mostly slightly worse than the results for the component-wise classification, despite achieving higher F-score for the joint classification task. This is simply due to the combined method tending to get both labels correct or both labels wrong, for a given post.

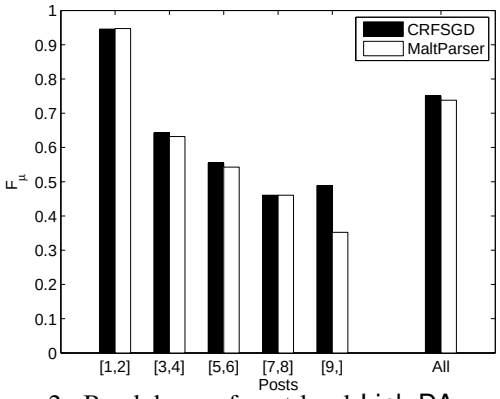


Figure 2: Breakdown of post-level Link-DA results for CRFSGD and MaltParser based on post position

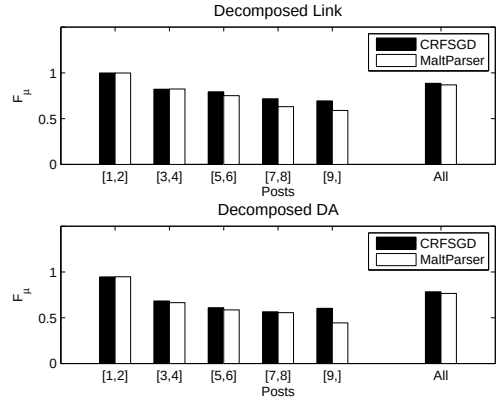


Figure 3: Breakdown of post-level Link and DA F-score based on the decomposition of CRFSGD and MaltParser classifications

6.2 Post Position-based Result Breakdown

One question in thread discourse structure classification is how accurate the predictions are at different depths in a thread (e.g. the first two posts vs. the second two posts). A breakdown of results across posts at different positions is presented in Figure 2.

The overall trend for both CRFSGD and MaltParser is that it becomes increasingly hard to classify posts as we continue through a thread, due to greater variability in discourse structure and greater sparsity in the data. However, it is interesting to note that the results for CRFSGD actually improve from posts 7 and 8 ($[7, 8]$) to posts 9 and onwards ($[9,]$). To further investigate this effect, we performed class decomposition over the joint classification predictions, and performed a similar breakdown of posts for Link and DA; the results are presented in Figure 3. It is clear that the anomaly for CRFSGD comes from the DA component, due to there being greater predictability in the dialogue for final posts in a thread (users tend to confirm a successful resolution of the problem, or report on successful external reproduction of the solution). MaltParser seems less adept at identifying that a post is at the end of a thread, and predicting the dialogue act accordingly. This observation is congruous with the findings of McDonald and Nivre (2007) that errors propagate, due to MaltParser's greedy inference strategy. The higher results for Link are to be expected, as throughout the thread, most posts tend to link locally.

Test \ B/down	$[1,2]$	$[1,4]$	$[1,6]$	$[1,8]$	$[All]$
$[1,2]$	.947/.947	—	—	—	—
$[1,4]$	.946/.947	.836/.841	—	—	
$[1,6]$	.946/.947	.840/.841	.800/.794	—	—
$[1,8]$	.946/.947	.840/.841	.800/.794	.780/.769	—
$[All]$	.946/.946	.840/.838	.800/.791	.776/.767	.756/.738

Table 5: Post-level Link-DA F-score for CRFSGD/MaltParser, based on in situ classification over sub-threads of different lengths (indicated in the rows), broken down over different post extents (indicated in the columns)

6.3 In Situ Structure Prediction

As described in Section 5.2, we simulate in situ thread discourse structure prediction by removing differing numbers of posts from the tail of the thread, and applying the trained model over the resultant sub-threads. The results for in situ classification are presented in Table 5, with the rows indicating the size of the test sub-thread, and the columns being a breakdown of results over different portions of the classified thread. The reason that we do not provide numbers for all cells in the table is that the size of the test sub-thread determines the post extents we can breakdown the results into, e.g. we cannot return results for posts 1–4 ($[1,4]$) when the size of the test thread was only two posts ($[1,2]$).

From the results, we can see that both CRFSGD and MaltParser are very robust when applied to partial threads, to the extent that we actually achieve higher results over shortened versions of the thread than over the complete thread in some instances, although the only difference that is statistically significant is over $[1,8]$ for CRFSGD, where the prediction over the partial thread is actually superior to that over the complete thread. From this, we can conclude that it is possible to apply our method to partial threads without any reduction in effectiveness relative to classification over complete threads. As such, our method is shown to be robust when applied to real-time analysis of dynamically evolving threads.

6.4 User profile feature analysis

In our experiments, we noticed that the user profile feature (UserProf) is the most effective feature for both CRFSGD and MaltParser. To gain a deeper insight into the behaviour of the feature, we binned the posts according to the number of times the author had posted in the training data, evaluated based on a user score ($uscore$) for each user:

$$uscore_i = \frac{\sum_{j=1}^{n_i} s_{p_{i,j}}}{n_i}$$

where n_i is the number of posts by user i, and $s_{p_{i,j}}$ is the number of posts by user i that occur as training instances for other posts by the same author. $uscore$ reflects the average training–test post ratio per user in cross-validation. Note that as we include all posts from a given thread in a single partition during cross-validation, it is possible for an author to have posted 4 times, but have a $uscore$ of 0 due to those posts all occurring in the same thread.

Bin	$uscore$	Posts per user	Total users	Total posts
High	224.6	251	1	251
Medium	1∼41.7	4∼48	45	395
Low	0	2∼4	157	377
Very Low	0	1	309	309

Table 6: Statistics for the 4 groups of users

We ranked the users in the dataset in descending order of $uscore$, sub-ranking on n_i in cases of a tie in $uscore$. The users were binned into 4 groups of roughly equal post size. The detailed statistics are shown in Table 6, noting that the high-frequency bin ("High") contains posts from a single user. We present the post-level micro-averaged F-score for posts in each bin based on CRFSGD, with and without user profile features, in Figure 4.

Contrary to expectation, the UserProf features have the greatest impact for users with fewer posts. In fact, a statistically significant difference was observed only for users with no posts in the training data ($uscore = 0$), where the F-score jumped over 10% in absolute terms for both the Low and Very Low bins. Our explanation for this effect is that the

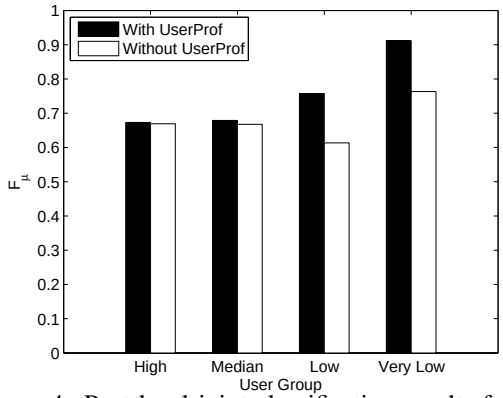


Figure 4: Post-level joint classification results for users binned by $uscore$, based on CRFSGD with and without UserProf features)

lack of user profile information is predictive of the sort of posts we can expect from a user (i.e. they tend to be newbie users, asking questions).

7 Conclusions and Future Work

In this research, we explored the joint classification of web user forum thread discourse structure, in the form of a rooted directed acyclic graph over posts, with edges labelled with dialogue acts. Three classification approaches were proposed: separately predicting Link and DA labels, and composing them into a joint class; predicting a combined Link-DA class using a structured classifier; and applying dependency parsing to the problem. We found the combined approach based on CRFSGD to perform best over the task, closely followed by dependency parsing with MaltParser.

We also examined the task of in situ classification of dialogue structure, in the form of predicting the discourse structure of partial threads, as contrasted with classifying only complete threads. We found that there was no drop in F-score over different subextents of the thread in classifying partial threads, despite the relative lack of thread context.

In future work, we plan to delve further into dependency parsing, looking specifically at the implications of multi-headedness and disconnected subgraphs on dependency parsing. We also intend to carry out meta-classification, combining the predictions of CRFSGD and MaltParser.

Our user profile features were found to be the pick of our features, but counter-intuitively, to benefit users with no posts in the training data, rather than prolific users. We wish to explore this effect further, including incorporating unsupervised user-level features into our classifiers.

Acknowledgements

The authors wish to acknowledge the development efforts of Johan Hall in configuring MaltParser to handle numeric features, and be able to parse thread structures. NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT Centre of Excellence programme.

References

Léon Bottou. 2011. CRFSGD software. `http://leon.bottou.org/projects/sgd`.

Xin Cao, Gao Cong, Bin Cui, Christian S. Jensen, and Ce Zhang. 2009. The use of categorization information in language models for question retrieval. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 265–274, Hong Kong, China.

Vitor R. Carvalho and William W. Cohen. 2005. On the collective classification of email "speech acts". In *Proceedings of 28th International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005)*, pages 345–352.

Jeffrey Chan and Conor Hayes. 2010. Decomposing discussion forums using user roles. In *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line (WebSci10)*, pages 1–8, Raleigh, USA.

Jeffrey Chan, Conor Hayes, and Elizabeth M. Daly. 2010. Decomposing discussion forums using user roles. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media (ICWSM 2010)*, pages 215–8, Washington, USA.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "speech acts". In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 309–316, Barcelona, Spain.

Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding question-answer

pairs from online forums. In *Proceedings of 31st International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08)*, pages 467–474, Singapore.

Daniel Dahlmeier, Hwee Tou Ng, and Tanja Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 450–458, Singapore. Association for Computational Linguistics.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. 2008. Using conditional random fields to extract context and answers of questions from online forums. In *Proceedings of the 46th Annual Meeting of the ACL: HLT (ACL 2008)*, pages 710–718, Columbus, USA.

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 30–41, Vancouver, Canada.

Jonathan L. Elsas and Jaime G. Carbonell. 2009. It pays to be picky: An evaluation of thread retrieval in online forums. In *Proceedings of 32nd International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*, pages 714–715, Boston, USA.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of the 46th Annual Meeting of the ACL: HLT (ACL 2008)*, pages 834–842, Columbus, USA.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2009)*, pages 326–334, Boulder, Colorado. Association for Computational Linguistics.

Blaz Fortuna, Eduarda Mendes Rodrigues, and Natasa Milic-Frayling. 2007. Improving the classification of newsgroup messages through social network analysis. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007)*, pages 877–880, Lisbon, Portugal.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intention and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Edward Ivanovic. 2008. Automatic instant messaging dialogue using statistical models and dialogue acts. Master's thesis, University of Melbourne.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010a. Classifying dialogue acts in one-on-one live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 862–871, Boston, USA.

Su Nam Kim, Li Wang, and Timothy Baldwin. 2010b. Tagging and linking web forum posts. In *Proceedings of the 14th Conference on Computational Natural Language Learning (CoNLL-2010)*, pages 192–202, Uppsala, Sweden.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–127.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, USA.

Andrew Lampert, Robert Dale, and Cécile Paris. 2008. The nature of requests and commitments in email messages. In *Proceedings of the AAAI 2008 Workshop on Enhanced Messaging*, pages 42–47, Chicago, USA.

Robert Leaman, Laura Wojtulewicz, Ryan Sullivan, Annie Skariah, Jian Yang, and Graciela Gonzalez. 2010. Towards internet-age pharmacovigilance: Extracting adverse drug reactions from user posts in health-related social networks. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing (ACL 2010)*, pages 117–125, Uppsala, Sweden.

Oliver Lemon, Alex Gruenstein, and Stanley Peters. 2002. Collaborative activities and multi-tasking in dialogue systems. *Traitement Automatique des Langues (TAL), Special Issue on Dialogue*, 43(2):131–154.

Chen Lin, Jiang-Ming Yang, Rui Cai, Xin-Jing Wang, Wei Wang, and Lei Zhang. 2009. Modeling semantics and structure of discussion threads. In *Proceedings of the 18th International Conference on the World Wide Web (WWW 2009)*, pages 1103–1104, Madrid, Spain.

Marco Lui and Timothy Baldwin. 2009. You are what you post: User-level features in threaded discourse. In *Proceedings of the 14th Australasian Document Computing Symposium (ADCS 2009)*, Sydney, Australia.

Marco Lui and Timothy Baldwin. 2010. Classifying user forum participants: Separating the gurus from the hacks, and other tales of the internet. In *Proceedings of the 2010 Australasian Language Technology Workshop (ALTW 2010)*, pages 49–57, Melbourne, Australia.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 122–131, Prague, Czech Republic.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of*

the European Chapter of the Association for Computational Linguistics (EACL 2006), pages 81–88, Trento, Italy.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, Canada.

Gabriel Murray, Steve Renals, Jean Carletta, and Johanna Moore. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 367–374.

Klemens Muthmann, Wojciech M. Barczyński, Falk Brauer, and Alexander Löser. 2009. Near-duplicate detection for web-forums. In *Proceedings of the 2009 International Database Engineering & Applications Symposium (IDEAS 2009)*, pages 142–151, Cetraro, Italy.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, France.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together (ACL-2004)*, pages 50–57, Barcelona, Spain.

Carolyn Penstein Rosé, Barbara Di Eugenio, Lori S. Levin, and Carol Van Ess-Dykema. 1995. Discourse processing of dialogues with multiple threads. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 31–38, Cambridge, USA.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 753–760, Manchester, UK.

Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT-09)*, pages 81–84, Paris, France.

Anne Schuth, Maarten Marx, and Maarten de Rijke. 2007. Extracting the discussion structure in comments on news-articles. In *Proceedings of the 9th Annual ACM International Workshop on Web Information and Data Management*, pages 97–104, Lisboa, Portugal.

Jangwon Seo, W. Bruce Croft, and David A. Smith. 2009. Online community search using thread structure. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 1907–1910, Hong Kong, China.

Elinzabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The ICSI meeting recorder dialog act (MRDA) corpus. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 97–100, Cambridge, USA.

Parikshit Sondhi, Manish Gupta, ChengXiang Zhai, and Julia Hockenmaier. 2010. Shallow information extraction from medical forum data. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010), Posters Volume*, pages 1158–1166, Beijing, China.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 149–156, Edmonton, Canada.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Pail Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 225–228, Ann Arbor, Michigan. Association for Computational Linguistics.

Pasi Tapanainen and Timo Jarvinen. 1997. A non-projective dependency parser. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 64–71, Washington, USA.

Nayer Wanas, Motaz El-Saban, Heba Ashour, and Waleed Ammar. 2008. Automatic scoring of online discussion posts. In *Proceeding of the 2nd ACM workshop on Information credibility on the web (WICOW '08)*, pages 19–26, Napa Valley, USA.

Yi-Chia Wang and Carolyn P. Rosé. 2010. Making conversational structure explicit: identification of initiation-response pairs within online discussions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*, pages 673–676.

Yi-Chia Wang, Mahesh Joshi, and Carolyn Rosé. 2007. A feature based approach to leveraging context for classifying newsgroup style discussion segments. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions (ACL 2007)*, pages 73–76, Prague, Czech Republic.

Yi-Chia Wang, Mahesh Joshi, William W. Cohen, and Carolyn Rosé. 2008. Recovering implicit thread structure in newsgroup style conversations. In *Proceedings of the Second International Conference on Weblogs and Social Media (ICWSM 2008)*, pages 152–160, Seattle, USA.

V. Warnke, R. Kompe, H. Niemann, and E. Nöth. 1997. Integrated dialog act segmentation and classification using prosodic features and language models. In *Proc. Eurospeech*, volume 1, pages 207–210.

Markus Weimer and Iryna Gurevych. 2007. Predicting the perceived quality of web forum posts. In *Proceedings of the 2007 International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, pages 643–648, Borovets, Bulgaria.

Markus Weimer, Iryna Gurevych, and Max Mühlhäuser. 2007. Automatically assessing the post quality in online discussions on software. In *Proceedings of the 45th Annual Meeting of the ACL: Interactive Poster and Demonstration Sessions*, pages 125–128, Prague, Czech Republic.

Armin Weinberger and Frank Fischer. 2006. A framework to analyze argumentative knowledge construction in computer-supported collaborative learning. *Computers & Education*, 46:71–95, January.

Florian Wolf and Edward Gibson. 2005. Representing discourse coherence: A corpus-based study. *Computational Linguistics*, 31(2):249–287.

Wensi Xi, Jesper Lind, and Eric Brill. 2004. Learning effective ranking functions for newsgroup search. In *Proceedings of 27th International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 394–401. Sheffield, UK.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 947–953, Saarbrücken, Germany.

Exact Decoding of Phrase-Based Translation Models through Lagrangian Relaxation

Yin-Wen Chang
MIT CSAIL
Cambridge, MA 02139, USA
yinwen@csail.mit.edu

Michael Collins
Department of Computer Science,
Columbia University,
New York, NY 10027, USA
mcollins@cs.columbia.edu

Abstract

This paper describes an algorithm for exact decoding of phrase-based translation models, based on Lagrangian relaxation. The method recovers exact solutions, with certificates of optimality, on over 99% of test examples. The method is much more efficient than approaches based on linear programming (LP) or integer linear programming (ILP) solvers: these methods are not feasible for anything other than short sentences. We compare our method to MOSES (Koehn et al., 2007), and give precise estimates of the number and magnitude of search errors that MOSES makes.

1 Introduction

Phrase-based models (Och et al., 1999; Koehn et al., 2003; Koehn et al., 2007) are a widely-used approach for statistical machine translation. The decoding problem for phrase-based models is NP-hard[1]; because of this, previous work has generally focused on approximate search methods, for example variants of beam search, for decoding.

This paper describes an algorithm for exact decoding of phrase-based models, based on Lagrangian relaxation (Lemaréchal, 2001). The core of the algorithm is a dynamic program for phrase-based translation which is efficient, but which allows some ill-formed translations. More specifically, the dynamic program searches over the space of translations where exactly N words are translated (N is the number of words in the source-language sentence), but where some source-language words may be translated zero times, or some source-language words may be translated more than once. Lagrangian relaxation is used to enforce the constraint that each source-language word should be translated exactly once. A subgradient algorithm is used to optimize the dual problem arising from the relaxation.

The first technical contribution of this paper is the basic Lagrangian relaxation algorithm. By the usual guarantees for Lagrangian relaxation, if this algorithm converges to a solution where all constraints are satisfied (i.e., where each word is translated exactly once), then the solution is guaranteed to be optimal. For some source-language sentences however, the underlying relaxation is loose, and the algorithm will not converge. The second technical contribution of this paper is a method that incrementally adds constraints to the underlying dynamic program, thereby tightening the relaxation until an exact solution is recovered.

We describe experiments on translation from German to English, using phrase-based models trained by MOSES (Koehn et al., 2007). The method recovers exact solutions, with certificates of optimality, on over 99% of test examples. On over 78% of examples, the method converges with zero added constraints (i.e., using the basic algorithm); 99.67% of all examples converge with 9 or fewer constraints. We compare to a linear programming (LP)/integer linear programming (ILP) based decoder. Our method is much more efficient: LP or ILP decoding is not feasible for anything other than short sentences,[2] whereas the average decoding time for our method (for sentences of length 1-50 words) is 121 seconds per sentence. We also compare our method to MOSES, and give precise estimates of the number and magnitude of search errors that MOSES makes. Even with large beam sizes, MOSES makes a significant number of search errors. As far as we are aware, previous work has not successfully re-

[1] We refer here to the phrase-based models of (Koehn et al., 2003; Koehn et al., 2007), considered in this paper. Other variants of phrase-based models, which allow polynomial time decoding, have been proposed, see the related work section.

[2] For example ILP decoding for sentences of lengths 11-15 words takes on average 2707.8 seconds.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 26–37,
Edinburgh, Scotland, UK, July 27–31, 2011.

covered exact solutions for the type of phrase-based models used in MOSES.

2 Related Work

Lagrangian relaxation is a classical technique for solving combinatorial optimization problems (Korte and Vygen, 2008; Lemaréchal, 2001). Dual decomposition, a special case of Lagrangian relaxation, has been applied to inference problems in NLP (Koo et al., 2010; Rush et al., 2010), and also to Markov random fields (Wainwright et al., 2005; Komodakis et al., 2007; Sontag et al., 2008). Earlier work on belief propagation (Smith and Eisner, 2008) is closely related to dual decomposition. Recently, Rush and Collins (2011) describe a Lagrangian relaxation algorithm for decoding for syntactic translation; the algorithmic construction described in the current paper is, however, very different in nature to this work.

Beam search stack decoders (Koehn et al., 2003) are the most commonly used decoding algorithm for phrase-based models. Dynamic-programming-based beam search algorithms are discussed for both word-based and phrase-based models by Tillmann and Ney (2003) and Tillmann (2006).

Several works attempt exact decoding, but efficiency remains an issue. Exact decoding via integer linear programming (ILP) for IBM model 4 (Brown et al., 1993) has been studied by Germann et al. (2001), with experiments using a bigram language model for sentences up to eight words in length. Riedel and Clarke (2009) have improved the efficiency of this work by using a cutting-plane algorithm, and experimented with sentence lengths up to 30 words (again with a bigram LM). Zaslavskiy et al. (2009) formulate the phrase-based decoding problem as a traveling salesman problem (TSP), and take advantage of existing exact and approximate approaches designed for TSP. Their translation experiment uses a bigram language model and applies an approximate algorithm for TSP. Och et al. (2001) propose an A* search algorithm for IBM model 4, and test on sentence lengths up to 14 words. Other work (Kumar and Byrne, 2005; Blackwood et al., 2009) has considered variants of phrase-based models with restrictions on reordering that allow exact, polynomial time decoding, using finite-state transducers.

The idea of incrementally adding constraints to tighten a relaxation until it is exact is a core idea in combinatorial optimization. Previous work on this topic in NLP or machine learning includes work on inference in Markov random fields (Sontag et al., 2008); work that encodes constraints using finite-state machines (Tromble and Eisner, 2006); and work on non-projective dependency parsing (Riedel and Clarke, 2006).

3 The Phrase-based Translation Model

This section establishes notation for phrase-based translation models, and gives a definition of the decoding problem. The phrase-based model we use is the same as that described by Koehn et al. (2003), as implemented in MOSES (Koehn et al., 2007).

The input to a phrase-based translation system is a source-language sentence with N words, $x_1 x_2 \dots x_N$. A *phrase table* is used to define the set of possible *phrases* for the sentence: each phrase is a tuple $p = (s, t, e)$, where (s, t) are indices representing a contiguous span in the source-language sentence (we have $s \leq t$), and e is a target-language string consisting of a sequence of target-language words. For example, the phrase $p = (2, 5, \textit{the dog})$ would specify that words $x_2 \dots x_5$ have a translation in the phrase table as "the dog". Each phrase p has a score $g(p) = g(s, t, e)$: this score will typically be calculated as a log-linear combination of features (e.g., see Koehn et al. (2003)).

We use $s(p)$, $t(p)$ and $e(p)$ to refer to the three components (s, t, e) of a phrase p.

The output from a phrase-based model is a sequence of phrases $y = \langle p_1 p_2 \dots p_L \rangle$. We will often refer to an output y as a *derivation*. The derivation y defines a target-language translation $e(y)$, which is formed by concatenating the strings $e(p_1), e(p_2), \dots, e(p_L)$. For two consecutive phrases $p_k = (s, t, e)$ and $p_{k+1} = (s', t', e')$, the *distortion distance* is defined as $\delta(t, s') = |t + 1 - s'|$. The score for a translation is then defined as

$$f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta \times \delta(t(p_k), s(p_{k+1}))$$

where $\eta \in \mathbb{R}$ is often referred to as the distortion penalty, and typically takes a negative value. The function $h(e(y))$ is the score of the string $e(y)$ under

a language model.[3]

The decoding problem is to find

$$\arg\max_{y \in \mathcal{Y}} f(y)$$

where $\mathcal{Y}$ is the set of valid derivations. The set $\mathcal{Y}$ can be defined as follows. First, for any derivation $y = \langle p_1 p_2 \ldots p_L \rangle$, define $y(i)$ to be the number of times that the source-language word x_i has been translated in y: that is, $y(i) = \sum_{k=1}^{L} [[s(p_k) \leq i \leq t(p_k)]]$, where $[[\pi]] = 1$ if π is true, and 0 otherwise. Then $\mathcal{Y}$ is defined as the set of finite length sequences $\langle p_1 p_2 \ldots p_L \rangle$ such that:

1. Each word in the input is translated exactly once: that is, $y(i) = 1$ for $i = 1 \ldots N$.
2. For each pair of consecutive phrases p_k, p_{k+1} for $k = 1 \ldots L-1$, we have $\delta(t(p_k), s(p_{k+1})) \leq d$, where d is the *distortion limit*.

An exact dynamic programming algorithm for this problem uses states (w_1, w_2, b, r), where (w_1, w_2) is a target-language bigram that the partial translation ended with, b is a bit-string denoting which source-language words have been translated, and r is the end position of the previous phrase (e.g., see Koehn et al. (2003)). The bigram (w_1, w_2) is needed for calculation of trigram language model scores; r is needed to enforce the distortion limit, and to calculate distortion costs. The bit-string b is needed to ensure that each word is translated exactly once. Since the number of possible bit-strings is exponential in the length of sentence, exhaustive dynamic programming is in general intractable. Instead, people commonly use heuristic search methods such as beam search for decoding. However, these methods have no guarantee of returning the highest scoring translation.

4 A Decoding Algorithm based on Lagrangian Relaxation

We now describe a decoding algorithm for phrase-based translation, based on Lagrangian relaxation. We first describe a dynamic program for decoding which is efficient, but which relaxes the $y(i) = 1$ constraints described in the previous section. We then describe the Lagrangian relaxation algorithm, which introduces Lagrange multipliers for each constraint of the form $y(i) = 1$, and uses a subgradient algorithm to minimize the dual arising from the relaxation. We conclude with theorems describing formal properties of the algorithm, and with an example run of the algorithm.

4.1 An Efficient Dynamic Program

As described in the previous section, our goal is to find the optimal translation $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$. We will approach this problem by defining a set $\mathcal{Y}'$ such that $\mathcal{Y} \subset \mathcal{Y}'$, and such that

$$\arg\max_{y \in \mathcal{Y}'} f(y)$$

can be found efficiently using dynamic programming. The set $\mathcal{Y}'$ omits some constraints—specifically, the constraints that each source-language word is translated once, i.e., that $y(i) = 1$ for $i = 1 \ldots N$—that are enforced for members of $\mathcal{Y}$. In the next section we describe how to reintroduce these constraints using Lagrangian relaxation. The set $\mathcal{Y}'$ does, however, include a looser constraint, namely that $\sum_{i=1}^{N} y(i) = N$, which requires that exactly N words are translated.

We now give the dynamic program that defines $\mathcal{Y}'$. The main idea will be to replace bit-strings (as described in the previous section) by a much smaller number of dynamic programming states. Specifically, the states of the new dynamic program will be tuples (w_1, w_2, n, l, m, r). The pair (w_1, w_2) is again a target-language bigram corresponding to the last two words in the partial translation, and the integer r is again the end position of the previous phrase. The integer n is the number of words that have been translated thus far in the dynamic programming algorithm. The integers l and m specify a contiguous span $x_l \ldots x_m$ in the source-language sentence; this span is the last contiguous span of words that have been translated thus far.

The dynamic program can be viewed as a shortest-path problem in a directed graph, with nodes in the graph corresponding to states (w_1, w_2, n, l, m, r). The transitions in the

[3] The language model score usually includes a word insertion score that controls the length of translations. The relative weights of the $g(p)$ and $h(e(y))$ terms, and the value for η, are typically chosen using MERT training (Och, 2003).

graph are defined as follows. For each state (w_1, w_2, n, l, m, r), we consider any phrase $p = (s, t, e)$ with $e = (e_0 \dots e_{M-1} e_M)$ such that: 1) $\delta(r, s) \leq d$; and 2) $t < l$ or $s > m$. The former condition states that the phrase should satisfy the distortion limit. The latter condition requires that there is no overlap of the new phrase's span (s, t) with the span (l, m). For any such phrase, we create a transition

$$(w_1, w_2, n, l, m, r) \xrightarrow{p=(s,t,e)} (w'_1, w'_2, n', l', m', r')$$

where

- $(w'_1, w'_2) = \begin{cases} (e_{M-1}, e_M) & \text{if } M \geq 2 \\ (w_2, e_1) & \text{if } M = 1 \end{cases}$
- $n' = n + t - s + 1$
- $(l', m') = \begin{cases} (l,\ t\ ) & \text{if } s = m + 1 \\ (s, m) & \text{if } t = l - 1 \\ (s,\ t\ ) & \text{otherwise} \end{cases}$
- $r' = t$

The new target-language bigram (w'_1, w'_2) is the last two words of the partial translation after including phrase p. It comes from either the last two words of e, or, if e consists of a single word, the last word of the previous bigram, w_2, and the first and only word, e_1, in e. (l', m') is expanded from (l, m) if the spans (l, m) and (s, t) are adjacent. Otherwise, (l', m') will be the same as (s, t).

The score of the transition is given by a sum of the phrase translation score $g(p)$, the language model score, and the distortion cost $\eta \times \delta(r, s)$. The trigram language model score is $h(e_1|w_1, w_2) + h(e_2|w_2, e_1) + \sum_{i=1}^{M-2} h(e_{i+2}|e_i, e_{i+1})$, where $h(w_3|w_1, w_2)$ is a trigram score (typically a log probability plus a word insertion score).

We also include start and end states in the directed graph. The start state is (<s>, <s>, 0, 0, 0, 0) where <s> is the start symbol in the language model. For each state (w_1, w_2, n, l, m, r), such that $n = N$, we create a transition to the end state. This transition takes the form

$$(w_1, w_2, N, l, m, r) \xrightarrow{(N, N+1, \text{</s>})} \text{END}$$

For this transition, we define the score as $score = h(\text{</s>}|w_1, w_2)$; thus this transition incorporates the end symbol </s> in the language model.

The states and transitions we have described form a directed graph, where each path from the start state to the end state corresponds to a sequence of phrases $p_1 p_2 \dots p_L$. We define $\mathcal{Y}'$ to be the full set of such sequences. We can use the Viterbi algorithm to solve $\arg\max_{y \in \mathcal{Y}'} f(y)$ by simply searching for the highest scoring path from the start state to the end state.

The set $\mathcal{Y}'$ clearly includes derivations that are ill-formed, in that they may include words that have been translated 0 times, or more than 1 time. The first line of Figure 2 shows one such derivation (corresponding to the translation *the quality and also the and the quality and also .*). For each phrase we show the English string (e.g., *the quality*) together with the span of the phrase (e.g., 3, 6). The values for $y(i)$ are also shown. It can be verified that this derivation is a valid member of $\mathcal{Y}'$. However, $y(i) \neq 1$ for several values of i: for example, words 1 and 2 are translated 0 times, while word 3 is translated twice.

Other dynamic programs, and definitions of $\mathcal{Y}'$, are possible: for example an alternative would be to use a dynamic program with states (w_1, w_2, n, r). However, including the previous contiguous span (l, m) makes the set $\mathcal{Y}'$ a closer approximation to $\mathcal{Y}$. In experiments we have found that including the previous span (l, m) in the dynamic program leads to faster convergence of the subgradient algorithm described in the next section, and in general to more stable results. This is in spite of the dynamic program being larger; it is no doubt due to $\mathcal{Y}'$ being a better approximation of $\mathcal{Y}$.

4.2 The Lagrangian Relaxation Algorithm

We now describe the Lagrangian relaxation decoding algorithm for the phrase-based model. Recall that in the previous section, we defined a set $\mathcal{Y}'$ that allowed efficient dynamic programming, and such that $\mathcal{Y} \subset \mathcal{Y}'$. It is easy to see that $\mathcal{Y} = \{y : y \in \mathcal{Y}', \text{ and } \forall i,\ y(i) = 1\}$. The original decoding problem can therefore be stated as:

$$\arg\max_{y \in \mathcal{Y}'} f(y) \text{ such that } \forall i,\ y(i) = 1$$

We use Lagrangian relaxation (Korte and Vygen, 2008) to deal with the $y(i) = 1$ constraints. We introduce Lagrange multipliers $u(i)$ for each such constraint. The Lagrange multipliers $u(i)$ can take any positive or negative value. The Lagrangian is

$$L(u, y) = f(y) + \sum_i u(i)(y(i) - 1)$$

Initialization: $u^0(i) \leftarrow 0 \quad$ for $i = 1 \ldots N$
for $t = 1 \ldots T$
  $y^t = \arg\max_{y \in \mathcal{Y}'} L(u^{t-1}, y)$
  if $y^t(i) = 1$ for $i = 1 \ldots N$
    return y^t
  else
    for $i = 1 \ldots N$
      $u^t(i) = u^{t-1}(i) - \alpha^t \left(y^t(i) - 1\right)$

Figure 1: The decoding algorithm. $\alpha^t > 0$ is the step size at the t'th iteration.

The dual objective is then

$$L(u) = \max_{y \in \mathcal{Y}'} L(u, y).$$

and the dual problem is to solve

$$\min_u L(u).$$

The next section gives a number of formal results describing how solving the dual problem will be useful in solving the original optimization problem.

We now describe an algorithm that solves the dual problem. By standard results for Lagrangian relaxation (Korte and Vygen, 2008), $L(u)$ is a convex function; it can be minimized by a subgradient method. If we define

$$y_u = \arg\max_{y \in \mathcal{Y}'} L(u, y)$$

and $\gamma_u(i) = y_u(i) - 1$ for $i = 1 \ldots N$, then γ_u is a subgradient of $L(u)$ at u. A subgradient method is an iterative method for minimizing $L(u)$, which perfoms updates $u^t \leftarrow u^{t-1} - \alpha^t \gamma_{u^{t-1}}$ where $\alpha^t > 0$ is the step size for the t'th subgradient step.

Figure 1 depicts the resulting algorithm. At each iteration, we solve

$$\arg\max_{y \in \mathcal{Y}'} \left(f(y) + \sum_i u(i)(y(i) - 1) \right)$$
$$= \arg\max_{y \in \mathcal{Y}'} \left(f(y) + \sum_i u(i) y(i) \right)$$

by the dynamic program described in the previous section. Incorporating the $\sum_i u(i) y(i)$ terms in the dynamic program is straightforward: we simply redefine the phrase scores as

$$g'(s, t, e) = g(s, t, e) + \sum_{i=s}^{t} u(i)$$

Intuitively, each Lagrange multiplier $u(i)$ penalizes or rewards phrases that translate word i; the algorithm attempts to adjust the Lagrange multipliers in such a way that each word is translated exactly once. The updates $u^t(i) = u^{t-1}(i) - \alpha^t(y^t(i) - 1)$ will decrease the value for $u(i)$ if $y^t(i) > 1$, increase the value for $u(i)$ if $y^t(i) = 0$, and leave $u(i)$ unchanged if $y^t(i) = 1$.

4.3 Properties

We now give some theorems stating formal properties of the Lagrangian relaxation algorithm. These results for Lagrangian relaxation are well known: for completeness, we state them here. First, define y^* to be the optimal solution for our original problem:

Definition 1. $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$

Our first theorem states that the dual function provides an upper bound on the score for the optimal translation, $f(y^*)$:

Theorem 1. *For any value of $u \in \mathbb{R}^N$, $L(u) \geq f(y^*)$.*

Proof.

$$\begin{aligned} L(u) &= \max_{y \in \mathcal{Y}'} \; f(y) + \sum_i u(i)(y(i) - 1) \\ &\geq \max_{y \in \mathcal{Y}} \; f(y) + \sum_i u(i)(y(i) - 1) \\ &= \max_{y \in \mathcal{Y}} \; f(y) \end{aligned}$$

The first inequality follows because $\mathcal{Y} \subset \mathcal{Y}'$. The final equality is true since any $y \in \mathcal{Y}$ has $y(i) = 1$ for all i, implying that $\sum_i u(i)(y(i) - 1) = 0$. □

The second theorem states that under an appropriate choice of the step sizes α^t, the method converges to the minimum of $L(u)$. Hence we will successfully find the tightest possible upper bound defined by the dual $L(u)$.

Theorem 2. *For any sequence $\alpha^1, \alpha^2, \ldots$ If 1) $\lim_{t \to \infty} \alpha^t \to 0$; 2) $\sum_{t=1}^{\infty} \alpha^t = \infty$, then $\lim_{t \to \infty} L(u^t) = \min_u L(u)$*

Proof. See Korte and Vygen (2008). □

Input German: dadurch können die qualität und die regelmäßige postzustellung auch weiterhin sichergestellt werden .

t	$L(u^{t-1})$	$y^t(i)$	derivation y^t
1	-10.0988	0 0 2 2 3 3 0 0 2 0 0 0 1	3, 6 the quality and | 9, 9 also | 6, 6 the | 5, 5 and | 3, 3 the | 4, 6 quality and | 9, 9 also | 13, 13 .
2	-11.1597	0 0 1 0 0 0 1 0 0 4 1 5 1	3, 3 the | 7, 7 regular | 12, 12 will | 10, 10 continue to | 12, 12 be | 10, 10 continue to | 12, 12 be | 10, 10 continue to | 12, 12 be | 10, 10 continue to | 11, 13 be guaranteed .
3	-12.3742	3 3 1 2 2 0 0 0 1 0 0 0 1	1, 2 in that way , | 5, 5 and | 2, 2 can | 1, 1 thus | 4, 4 quality | 1, 2 in that way , | 3, 5 the quality and | 9, 9 also | 13, 13 .
4	-11.8623	0 1 0 0 0 1 1 3 3 0 3 0 1	2, 2 can | 6, 7 the regular | 8, 8 distribution should | 9, 9 also | 11, 11 ensure | 8, 8 distribution should | 9, 9 also | 11, 11 ensure | 8, 8 distribution should | 9, 9 also | 11, 11 ensure | 13, 13 .
5	-13.9916	0 0 1 1 3 2 4 0 0 0 1 0 1	3, 3 the | 7, 7 regular | 5, 5 and | 7, 7 regular | 5, 5 and | 7, 7 regular | 6, 6 the | 4, 4 quality | 5, 7 and the regular | 11, 11 ensured | 13, 13 .
6	-15.6558	1 1 1 2 0 2 0 1 1 1 1 1 1	1, 2 in that way , | 3, 4 the quality of | 6, 6 the | 4, 4 quality of | 6, 6 the | 8, 8 distribution should | 9, 10 continue to | 11, 13 be guaranteed .
7	-16.1022	1 1 1 1 1 1 1 1 1 1 1 1 1	1, 2 in that way , | 3, 4 the quality | 5, 7 and the regular | 8, 8 distribution should | 9, 10 continue to | 11, 13 be guaranteed .

Figure 2: An example run of the algorithm in Figure 1. For each value of t we show the dual value $L(u^{t-1})$, the derivation y^t, and the number of times each word is translated, $y^t(i)$ for $i = 1 \dots N$. For each phrase in a derivation we show the English string e, together with the span (s, t): for example, the first phrase in the first derivation has English string *the quality and*, and span $(3, 6)$. At iteration 7 we have $y^t(i) = 1$ for $i = 1 \dots N$, and the translation is returned, with a guarantee that it is optimal.

Our final theorem states that if at any iteration the algorithm finds a solution y^t such that $y^t(i) = 1$ for $i = 1 \dots N$, then this is guaranteed to be the optimal solution to our original problem. First, define

Definition 2. $y_u = \arg\max_{y \in \mathcal{Y}'} L(u, y)$.

We then have the theorem

Theorem 3. *If $\exists\ u$, s.t. $y_u(i) = 1$ for $i = 1 \dots N$, then $f(y_u) = f(y^*)$, i.e. y_u is optimal.*

Proof. We have

$$\begin{aligned} L(u) &= \max_{y \in \mathcal{Y}'} \quad f(y) + \sum_i u(i)(y(i) - 1) \\ &= f(y_u) + \sum_i u(i)(y_u(i) - 1) \\ &= f(y_u) \end{aligned}$$

The second equality is true because of the definition of y_u. The third equality follows because by assumption $y_u(i) = 1$ for $i = 1 \dots N$. Because $L(u) = f(y_u)$ and $L(u) \geq f(y^*)$ for all u, we have $f(y_u) \geq f(y^*)$. But $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$, and $y_u \in \mathcal{Y}$, hence we must also have $f(y_u) \leq f(y^*)$. It follows that $f(y_u) = f(y^*)$. □

In some cases, however, the algorithm in Figure 1 may not return a solution y^t such that $y^t(i) = 1$ for all i. There could be two reasons for this. In the first case, we may not have run the algorithm for enough iterations T to see convergence. In the second case, the underlying relaxation may not be tight, in that there may not be *any* settings u for the Lagrange multipliers such that $y_u(i) = 1$ for all i.

Section 5 describes a method for tightening the underlying relaxation by introducing hard constraints (of the form $y(i) = 1$ for selected values of i). We will see that this method is highly effective in tightening the relaxation until the algorithm converges to an optimal solution.

4.4 An Example of the Algorithm

Figure 2 shows an example of how the algorithm works when translating a German sentence into an English sentence. After the first iteration, there are words that have been translated two or three times, and words that have not been translated. At each iteration, the Lagrangian multipliers are updated to encourage each word to be translated once. On this example, the algorithm converges to a solution where all words are translated exactly once, and the solution is guaranteed to be optimal.

5 Tightening the Relaxation

In some cases the algorithm in Figure 1 will not converge to $y(i) = 1$ for $i = 1 \dots N$ because the underlying relaxation is not tight. We now describe a method that incrementally tightens the Lagrangian relaxation algorithm until it provides an exact answer. In cases that do not converge, we introduce hard constraints to force certain words to be translated exactly once in the dynamic programming solver. In experiments we show that typically only a

$Optimize(\mathcal{C}, u)$
while (dual value still improving)
$y^* = \arg\max_{y \in \mathcal{Y}'_{\mathcal{C}}} L(u, y)$
if $y^*(i) = 1$ for $i = 1 \ldots N$ **return** y^*
else for $i = 1 \ldots N$
$u(i) = u(i) - \alpha\,(y^*(i) - 1)$
$count(i) = 0$ for $i = 1 \ldots N$
for $k = 1 \ldots K$
$y^* = \arg\max_{y \in \mathcal{Y}'_{\mathcal{C}}} L(u, y)$
if $y^*(i) = 1$ for $i = 1 \ldots N$ **return** y^*
else for $i = 1 \ldots N$
$u(i) = u(i) - \alpha\,(y^*(i) - 1)$
$count(i) = count(i) + [[y^*(i) \neq 1]]$
Let $\mathcal{C}' =$ set of G i's that have the largest value for $count(i)$, that are not in $\mathcal{C}$, and that are not adjacent to each other
return $Optimize(\mathcal{C} \cup \mathcal{C}', u)$

Figure 3: A decoding algorithm with incremental addition of constraints. The function $Optimize(\mathcal{C}, u)$ is a recursive function, which takes as input a set of constraints $\mathcal{C}$, and a vector of Lagrange multipliers, u. The initial call to the algorithm is with $\mathcal{C} = \emptyset$, and $u = 0$. $\alpha > 0$ is the step size. In our experiments, the step size decreases each time the dual value increases from one iteration to the next; see Appendix A.

few constraints are necessary.

Given a set $\mathcal{C} \subseteq \{1, 2, \ldots, N\}$, we define

$$\mathcal{Y}'_{\mathcal{C}} = \{y : y \in \mathcal{Y}', \text{ and } \forall\, i \in \mathcal{C},\ \ y(i) = 1\}$$

Thus $\mathcal{Y}'_{\mathcal{C}}$ is a subset of $\mathcal{Y}'$, formed by adding hard constraints of the form $y(i) = 1$ to $\mathcal{Y}'$. Note that $\mathcal{Y}'_{\mathcal{C}}$ remains as a superset of $\mathcal{Y}$, which enforces $y(i) = 1$ for all i. Finding $\arg\max_{y \in \mathcal{Y}'_{\mathcal{C}}} f(y)$ can again be achieved using dynamic programming, with the number of dynamic programming states increased by a factor of $2^{|\mathcal{C}|}$: dynamic programming states of the form (w_1, w_2, n, l, m, r) are replaced by states $(w_1, w_2, n, l, m, r, b_{\mathcal{C}})$ where $b_{\mathcal{C}}$ is a bit-string of length $|\mathcal{C}|$, which records which words in the set $\mathcal{C}$ have or haven't been translated in a hypothesis (partial derivation). Note that if $\mathcal{C} = \{1 \ldots N\}$, we have $\mathcal{Y}'_{\mathcal{C}} = \mathcal{Y}$, and the dynamic program will correspond to exhaustive dynamic programming.

We can again run a Lagrangian relaxation algorithm, using the set $\mathcal{Y}'_{\mathcal{C}}$ in place of $\mathcal{Y}'$. We will use Lagrange multipliers $u(i)$ to enforce the constraints $y(i) = 1$ for $i \notin \mathcal{C}$. Our goal will be to find a small set of constraints $\mathcal{C}$, such that Lagrangian relaxation will successfully recover an optimal solution. We will do this by incrementally adding elements to $\mathcal{C}$; that is, by incrementally adding constraints that tighten the relaxation.

The intuition behind our approach is as follows. Say we run the original algorithm, with the set $\mathcal{Y}'$, for several iterations, so that $L(u)$ is close to convergence (i.e., $L(u)$ is close to its minimal value). However, assume that we have not yet generated a solution y^t such that $y^t(i) = 1$ for all i. In this case we have some evidence that the relaxation may not be tight, and that we need to add some constraints. The question is, which constraints to add? To answer this question, we run the subgradient algorithm for K more iterations (e.g., $K = 10$), and at each iteration track which constraints of the form $y(i) = 1$ are violated. We then choose $\mathcal{C}$ to be the G constraints (e.g., $G = 3$) that are violated most often during the K additional iterations, and are not adjacent to each other. We recursively call the algorithm, replacing $\mathcal{Y}'$ by $\mathcal{Y}'_{\mathcal{C}}$; the recursive call may then return an exact solution, or alternatively again add more constraints and make a recursive call.[4]

Figure 3 depicts the resulting algorithm. We initially make a call to the algorithm $Optimize(\mathcal{C}, u)$ with $\mathcal{C}$ equal to the empty set (i.e., no hard constraints), and with $u(i) = 0$ for all i. In an initial phase the algorithm runs subgradient steps, while the dual is still improving. In a second step, if a solution has not been found, the algorithm runs for K more iterations, thereby choosing G additional constraints, then recursing.

If at any stage the algorithm finds a solution y^* such that $y^*(i) = 1$ for all i, then this is the solution to our original problem, $\arg\max_{y \in \mathcal{Y}} f(y)$. This follows because for any $\mathcal{C} \subseteq \{1 \ldots N\}$ we have $\mathcal{Y} \subseteq \mathcal{Y}'_{\mathcal{C}}$; hence the theorems in section 4.3 go through for $\mathcal{Y}'_{\mathcal{C}}$ in place of $\mathcal{Y}'$, with trivial modifications. Note also that the algorithm is guaranteed to eventually find the optimal solution, because eventually $\mathcal{C} = \{1 \ldots N\}$, and $\mathcal{Y} = \mathcal{Y}'_{\mathcal{C}}$.

[4] Formal justification for the method comes from the relationship between Lagrangian relaxation and linear programming relaxations. In cases where the relaxation is not tight, the subgradient method will essentially move between solutions whose convex combination form a fractional solution to an underlying LP relaxation (Nedić and Ozdaglar, 2009). Our method eliminates the fractional solution through the introduction of hard constraints.

# iter.	1-10 words	11-20 words	21-30 words	31-40 words	41-50 words	All sentences	
0-7	166 (89.7 %)	219 (39.2 %)	34 (6.0 %)	2 (0.6 %)	0 (0.0 %)	421 (23.1 %)	23.1 %
8-15	17 (9.2 %)	187 (33.5 %)	161 (28.4 %)	30 (8.6 %)	3 (1.8 %)	398 (21.8 %)	44.9 %
16-30	1 (0.5 %)	93 (16.7 %)	208 (36.7 %)	112 (32.3 %)	22 (13.1 %)	436 (23.9 %)	68.8 %
31-60	1 (0.5 %)	52 (9.3 %)	105 (18.6 %)	99 (28.5 %)	62 (36.9 %)	319 (17.5 %)	86.3 %
61-120	0 (0.0 %)	7 (1.3 %)	54 (9.5 %)	89 (25.6 %)	45 (26.8 %)	195 (10.7 %)	97.0 %
121-250	0 (0.0 %)	0 (0.0 %)	4 (0.7 %)	14 (4.0 %)	31 (18.5 %)	49 (2.7 %)	99.7 %
x	0 (0.0 %)	0 (0.0 %)	0 (0.0 %)	1 (0.3 %)	5 (3.0 %)	6 (0.3 %)	100.0 %

Table 1: Table showing the number of iterations taken for the algorithm to converge. x indicates sentences that fail to converge after 250 iterations. 97% of the examples converge within 120 iterations.

# cons.	1-10 words	11-20 words	21-30 words	31-40 words	41-50 words	All sentences	
0-0	183 (98.9 %)	511 (91.6 %)	438 (77.4 %)	222 (64.0 %)	82 (48.8 %)	1,436 (78.7 %)	78.7 %
1-3	2 (1.1 %)	45 (8.1 %)	94 (16.6 %)	87 (25.1 %)	50 (29.8 %)	278 (15.2 %)	94.0 %
4-6	0 (0.0 %)	2 (0.4 %)	27 (4.8 %)	24 (6.9 %)	19 (11.3 %)	72 (3.9 %)	97.9 %
7-9	0 (0.0 %)	0 (0.0 %)	7 (1.2 %)	13 (3.7 %)	12 (7.1 %)	32 (1.8 %)	99.7 %
x	0 (0.0 %)	0 (0.0 %)	0 (0.0 %)	1 (0.3 %)	5 (3.0 %)	6 (0.3 %)	100.0 %

Table 2: Table showing the number of constraints added before convergence of the algorithm in Figure 3, broken down by sentence length. Note that a maximum of 3 constraints are added at each recursive call, but that fewer than 3 constraints are added in cases where fewer than 3 constraints have $count(i) > 0$. x indicates the sentences that fail to converge after 250 iterations. 78.7% of the examples converge without adding any constraints.

The remaining question concerns the "dual still improving" condition; i.e., how to determine that the first phase of the algorithm should terminate. We do this by recording the first and second best dual values $L(u')$ and $L(u'')$ in the sequence of Lagrange multipliers $u^1, u^2, \ldots$ generated by the algorithm. Suppose that $L(u'')$ first occurs at iteration t''. If $\frac{L(u')-L(u'')}{t-t''} < \epsilon$, we say that the dual value does not decrease enough. The value for ϵ is a parameter of the approach: in experiments we used $\epsilon = 0.002$.

See the supplementary material for this submission for an example run of the algorithm.

When $\mathcal{C} \neq \emptyset$, A* search can be used for decoding, with the dynamic program for $\mathcal{Y}'$ providing admissible estimates for the dynamic program for $\mathcal{Y}'_{\mathcal{C}}$. Experiments show that A* gives significant improvements in efficiency. The supplementary material contains a full description of the A* algorithm.

6 Experiments

In this section, we present experimental results to demonstrate the efficiency of the decoding algorithm. We compare to MOSES (Koehn et al., 2007), a phrase-based decoder using beam search, and to a general purpose integer linear programming (ILP) solver, which solves the problem exactly.

The experiments focus on translation from German to English, using the Europarl data (Koehn, 2005). We tested on 1,824 sentences of length at most 50 words. The experiments use the algorithm shown in Figure 3. We limit the algorithm to a maximum of 250 iterations and a maximum of 9 hard constraints. The distortion limit d is set to be four, and we prune the phrase translation table to have 10 English phrases per German phrase.

Our method finds exact solutions on 1,818 out of 1,824 sentences (99.67%). (6 examples do not converge within 250 iterations.) Table 1 shows the number of iterations required for convergence, and Table 2 shows the number of constraints required for convergence, broken down by sentence length. In 1,436/1,818 (78.7%) sentences, the method converges without adding hard constraints to tighten the relaxation. For sentences with 1-10 words, the vast majority (183 out of 185 examples) converge with 0 constraints added. As sentences get longer, more constraints are often required. However most examples converge with 9 or fewer constraints.

Table 3 shows the average times for decoding, broken down by sentence length, and by the number of constraints that are added. As expected, decoding times increase as the length of sentences, and the number of constraints required, increase. The average run time across all sentences is 120.9 seconds. Table 3 also shows the run time of the method without the A* algorithm for decoding. The A* algorithm gives significant reductions in runtime.

# cons.	1-10 words		11-20 words		21-30 words		31-40 words		41-50 words		All sentences	
	A*	w/o	A*	w/o	A*	w/o	A*	w/o	A*	w/o	A*	w/o
0-0	0.8	0.8	9.7	10.7	47.0	53.7	153.6	178.6	402.6	492.4	64.6	76.1
1-3	2.4	2.9	23.2	28.0	80.9	102.3	277.4	360.8	686.0	877.7	241.3	309.7
4-6	0.0	0.0	28.2	38.8	111.7	163.7	309.5	575.2	1,552.8	1,709.2	555.6	699.5
7-9	0.0	0.0	0.0	0.0	166.1	500.4	361.0	1,467.6	1,167.2	3,222.4	620.7	1,914.1
mean	0.8	0.9	10.9	12.3	57.2	72.6	203.4	299.2	679.9	953.4	120.9	168.9
median	0.7	0.7	8.9	9.9	48.3	54.6	169.7	202.6	484.0	606.5	35.2	40.0

Table 3: The average time (in seconds) for decoding using the algorithm in Figure 3, with and without A* algorithm, broken down by sentence length and the number of constraints that are added. A* indicates speeding up using A* search; w/o denotes without using A*.

method		ILP		LP		
set	length	mean	median	mean	median	% frac.
$\mathcal{Y}''$	1-10	275.2	132.9	10.9	4.4	12.4 %
	11-15	2,707.8	1,138.5	177.4	66.1	40.8 %
	16-20	20,583.1	3,692.6	1,374.6	637.0	59.7 %
$\mathcal{Y}'$	1-10	257.2	157.7	18.4	8.9	1.1 %
	11-15	3607.3	1838.7	476.8	161.1	3.0 %

Table 4: Average and median time of the LP/ILP solver (in seconds). % frac. indicates how often the LP gives a fractional answer. $\mathcal{Y}'$ indicates the dynamic program using set $\mathcal{Y}'$ as defined in Section 4.1, and $\mathcal{Y}''$ indicates the dynamic program using states (w_1, w_2, n, r). The statistics for ILP for length 16-20 are based on 50 sentences.

6.1 Comparison to an LP/ILP solver

To compare to a linear programming (LP) or integer linear programming (ILP) solver, we can implement the dynamic program (search over the set $\mathcal{Y}'$) through linear constraints, with a linear objective. The $y(i) = 1$ constraints are also linear. Hence we can encode our relaxation within an LP or ILP. Having done this, we tested the resulting LP or ILP using Gurobi, a high-performance commercial grade solver. We also compare to an LP or ILP where the dynamic program makes use of states (w_1, w_2, n, r)—i.e., the span (l, m) is dropped, making the dynamic program smaller. Table 4 shows the average time taken by the LP/ILP solver. Both the LP and the ILP require very long running times on these shorter sentences, and running times on longer sentences are prohibitive. Our algorithm is more efficient because it leverages the structure of the problem, by directly using a combinatorial algorithm (dynamic programming).

6.2 Comparison to MOSES

We now describe comparisons to the phrase-based decoder implemented in MOSES. MOSES uses beam search to find approximate solutions.

The distortion limit described in section 3 is the same as that in Koehn et al. (2003), and is the same as that described in the user manual for MOSES (Koehn et al., 2007). However, a complicating factor for our comparisons is that MOSES uses an additional distortion constraint, not documented in the manual, which we describe here.[5] We call this constraint the *gap constraint*. We will show in experiments that without the gap constraint, MOSES fails to produce translations on many examples. In our experiments we will compare to MOSES both with and without the gap constraint (in the latter case, we discard examples where MOSES fails).

We now describe the gap constraint. For a sequence of phrases $p_1, \ldots, p_k$ define $\theta(p_1 \ldots p_k)$ to be the index of the left-most source-language word not translated in this sequence. For example, if the bit-string for $p_1 \ldots p_k$ is 111001101000, then $\theta(p_1 \ldots p_k) = 4$. A sequence of phrases $p_1 \ldots p_L$ satisfies the gap constraint if and only if for $k = 2 \ldots L$, $|t(p_k) + 1 - \theta(p_1 \ldots p_k)| \leq d$, where d is the distortion limit. We will call MOSES without this restriction MOSES-nogc, and MOSES with this restriction MOSES-gc.

Results for MOSES-nogc Table 5 shows the number of examples where MOSES-nogc fails to give a translation, and the number of search errors for those cases where it does give a translation, for a range of beam sizes. A search error is defined as a case where our algorithm produces an exact solution that has higher score than the output from MOSES-nogc. The number of search errors is significant, even for large beam sizes.

[5] Personal communication from Philipp Koehn; see also the software for MOSES.

Beam size	Fails	# search errors	percentage
100	650/1,818	214/1,168	18.32 %
200	531/1,818	207/1,287	16.08 %
1000	342/1,818	115/1,476	7.79 %
10000	169/1,818	68/1,649	4.12 %

Table 5: Table showing the number of examples where MOSES-nogc fails to give a translation, and the number/percentage of search errors for cases where it does give a translation.

Diff.	MOSES-gc s =100	MOSES-gc s =200	MOSES-nogc s=1000
0.000 – 0.125	66 (24.26%)	65 (24.07%)	32 (27.83%)
0.125 – 0.250	59 (21.69%)	58 (21.48%)	25 (21.74%)
0.250 – 0.500	65 (23.90%)	65 (24.07%)	25 (21.74%)
0.500 – 1.000	49 (18.01%)	49 (18.15%)	23 (20.00%)
1.000 – 2.000	31 (11.40%)	31 (11.48%)	5 (4.35%)
2.000 – 4.000	2 (0.74%)	2 (0.74%)	3 (2.61%)
4.000 –13.000	0 (0.00%)	0 (0.00%)	2 (1.74%)

Table 6: Table showing statistics for the difference between the translation score from MOSES, and from the optimal derivation, for those sentences where a search error is made. For MOSES-gc we include cases where the translation produced by our system is not reachable by MOSES-gc. The average score of the optimal derivations is -23.4.

Results for MOSES-gc MOSES-gc uses the gap constraint, and thus in some cases our decoder will produce derivations which MOSES-gc cannot reach. Among the 1,818 sentences where we produce a solution, there are 270 such derivations. For the remaining 1,548 sentences, MOSES-gc makes search errors on 2 sentences (0.13%) when the beam size is 100, and no search errors when the beam size is 200, 1,000, or 10,000.

Table 6 shows statistics for the magnitude of the search errors that MOSES-gc and MOSES-nogc make.

BLEU Scores Finally, table 7 gives BLEU scores (Papineni et al., 2002) for decoding using MOSES and our method. The BLEU scores under the two decoders are almost identical; hence while MOSES makes a significant proportion of search errors, these search errors appear to be benign in terms of their impact on BLEU scores, at least for this particular translation model. Future work should investigate why this is the case, and whether this applies to other models and language pairs.

7 Conclusions

We have described an exact decoding algorithm for phrase-based translation models, using Lagrangian relaxation. The algorithmic construction we have described may also be useful in other areas of NLP, for example natural language generation. Possible extensions to the approach include methods that incorporate the Lagrangian relaxation formulation within learning algorithms for statistical MT: we see this as an interesting avenue for future research.

type of Moses	beam size	# sents	Moses	our method
MOSES-gc	100	1,818	24.4773	24.5395
	200	1,818	24.4765	24.5395
	1,000	1,818	24.4765	24.5395
	10,000	1,818	24.4765	24.5395
MOSES-nogc	100	1,168	27.3546	27.3249
	200	1,287	27.0591	26.9907
	1,000	1,476	26.5734	26.6128
	10,000	1,649	25.6531	25.6620

Table 7: BLEU score comparisons. We consider only those sentences where both decoders produce a translation.

A Step Size

Similar to Koo et al. (2010), we set the step size at the t'th iteration to be $\alpha^t = 1/(1 + \lambda^t)$, where λ^t is the number of times that $L(u^{(t')}) > L(u^{(t'-1)})$ for all $t' \leq t$. Thus the step size decreases each time the dual value increases from one iteration to the next.

Acknowledgments Yin-Wen Chang and Michael Collins were supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022. Michael Collins was also supported by NSF grant IIS-0915176.

References

Graeme Blackwood, Adrià de Gispert, Jamie Brunning, and William Byrne. 2009. Large-scale statistical machine translation with weighted finite state transducers. In *Proceeding of the 2009 conference on Finite-State Methods and Natural Language Processing: Post-proceedings of the 7th International Workshop FSMNLP 2008*, pages 39–49, Amsterdam, The Netherlands, The Netherlands. IOS Press.

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

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceed-*

ings of the 39th Annual Meeting on Association for Computational Linguistics, ACL '01, pages 228–235.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL '03, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT Summit*.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Proceedings of the 11th International Conference on Computer Vision*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA, October. Association for Computational Linguistics.

Bernhard Korte and Jens Vygen. 2008. *Combinatorial Optimization: Theory and Application*. Springer Verlag.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 161–168.

Claude Lemaréchal. 2001. Lagrangian Relaxation. In *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions [based on a Spring School]*, pages 112–156, London, UK. Springer-Verlag.

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

Franz Josef Och, Christoph Tillmann, Hermann Ney, and Lehrstuhl Fiir Informatik. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *Proceedings of the workshop on Data-driven methods in machine translation - Volume 14*, DMMT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, ACL '03, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 129–137, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Riedel and James Clarke. 2009. Revisiting optimal decoding for machine translation IBM model 4. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 5–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through Lagrangian relaxation. In *Proceedings of ACL*.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Cambridge, MA, October. Association for Computational Linguistics.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 145–156.

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. 2008. Tightening LP relaxations for MAP using message passing. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 503–510.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29:97–133, March.

Christoph Tillmann. 2006. Efficient dynamic programming search algorithms for phrase-based SMT.

In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, CHSLP '06, pages 9–16.

Roy W. Tromble and Jason Eisner. 2006. A fast finite-state relaxation method for enforcing global constraints on sequence decoding. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martin Wainwright, Tommi Jaakkola, and Alan Willsky. 2005. MAP estimation via agreement on trees: Message-passing and linear programming. In *IEEE Transactions on Information Theory*, volume 51, pages 3697–3717.

Mikhail Zaslavskiy, Marc Dymetman, and Nicola Cancedda. 2009. Phrase-based statistical machine translation as a traveling salesman problem. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 333–341, Stroudsburg, PA, USA. Association for Computational Linguistics.

Optimal Search for Minimum Error Rate Training

Michel Galley
Microsoft Research
Redmond, WA 98052, USA
mgalley@microsoft.com

Chris Quirk
Microsoft Research
Redmond, WA 98052, USA
chrisq@microsoft.com

Abstract

Minimum error rate training is a crucial component to many state-of-the-art NLP applications, such as machine translation and speech recognition. However, common evaluation functions such as BLEU or word error rate are generally highly non-convex and thus prone to search errors. In this paper, we present LP-MERT, an exact search algorithm for minimum error rate training that reaches the global optimum using a series of reductions to linear programming. Given a set of N-best lists produced from S input sentences, this algorithm finds a linear model that is globally optimal with respect to this set. We find that this algorithm is polynomial in N and in the size of the model, but exponential in S. We present extensions of this work that let us scale to reasonably large tuning sets (e.g., one thousand sentences), by either searching only promising regions of the parameter space, or by using a variant of LP-MERT that relies on a beam-search approximation. Experimental results show improvements over the standard Och algorithm.

1 Introduction

Minimum error rate training (MERT)—also known as direct loss minimization in machine learning—is a crucial component in many complex natural language applications such as speech recognition (Chou et al., 1993; Stolcke et al., 1997; Juang et al., 1997), statistical machine translation (Och, 2003; Smith and Eisner, 2006; Duh and Kirchhoff, 2008; Chiang et al., 2008), dependency parsing (McDonald et al., 2005), summarization (McDonald, 2006), and phonetic alignment (McAllester et al., 2010). MERT directly optimizes the evaluation metric under which systems are being evaluated, yielding superior performance (Och, 2003) when compared to a likelihood-based discriminative method (Och and Ney, 2002). In complex text generation tasks like SMT, the ability to optimize BLEU (Papineni et al., 2001), TER (Snover et al., 2006), and other evaluation metrics is critical, since these metrics measure qualities (such as fluency and adequacy) that often do not correlate well with task-agnostic loss functions such as log-loss.

While competitive in practice, MERT faces several challenges, the most significant of which is search. The unsmoothed error count is a highly non-convex objective function and therefore difficult to optimize directly; prior work offers no algorithm with a good approximation guarantee. While much of the earlier work in MERT (Chou et al., 1993; Juang et al., 1997) relies on standard convex optimization techniques applied to non-convex problems, the Och algorithm (Och, 2003) represents a significant advance for MERT since it applies a series of special line minimizations that happen to be exhaustive and efficient. Since this algorithm remains inexact in the multidimensional case, much of the recent work on MERT has focused on extending Och's algorithm to find better search directions and starting points (Cer et al., 2008; Moore and Quirk, 2008), and on experimenting with other derivative-free methods such as the Nelder-Mead simplex algorithm (Nelder and Mead, 1965; Zens et al., 2007; Zhao and Chen, 2009).

In this paper, we present LP-MERT, an exact search algorithm for N-best optimization that exploits general assumptions commonly made with MERT, e.g., that the error metric is decomposable by sentence.[1] While there is no known optimal algo-

[1] Note that MERT makes two types of approximations. First, the set of all possible outputs is represented only approximately, by N-best lists, lattices, or hypergraphs. Second, error functions on such representations are non-convex and previous work only offers approximate techniques to optimize them. Our work avoids the second approximation, while the first one is unavoidable when optimization and decoding occur in distinct steps.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 38–49,
Edinburgh, Scotland, UK, July 27–31, 2011.

rithm to optimize general non-convex functions, the unsmoothed error surface has a special property that enables exact search: the set of translations produced by an SMT system for a given input is finite, so the piecewise-constant error surface contains only a *finite* number of constant regions. As in Och (2003), one could imagine exhaustively enumerating all constant regions and finally return the best scoring one—Och does this efficiently with each one-dimensional search—but the idea doesn't quite scale when searching all dimensions at once. Instead, LP-MERT exploits algorithmic devices such as lazy enumeration, divide-and-conquer, and linear programming to efficiently discard partial solutions that cannot be maximized by any linear model. Our experiments with thousands of searches show that LP-MERT is never worse than the Och algorithm, which provides strong evidence that our algorithm is indeed exact. In the appendix, we formally prove that this search algorithm is optimal. We show that this algorithm is polynomial in N and in the size of the model, but exponential in the number of tuning sentences. To handle reasonably large tuning sets, we present two modifications of LP-MERT that either search only promising regions of the parameter space, or that rely on a beam-search approximation. The latter modification copes with tuning sets of one thousand sentences or more, and outperforms the Och algorithm on a WMT 2010 evaluation task.

This paper makes the following contributions. To our knowledge, it is the first known exact search algorithm for optimizing task loss on N-best lists in general dimensions. We also present an approximate version of LP-MERT that offers a natural means of trading speed for accuracy, as we are guaranteed to eventually find the global optimum as we gradually increase beam size. This trade-off may be beneficial in commercial settings and in large-scale evaluations like the NIST evaluation, i.e., when one has a stable system and is willing to let MERT run for days or weeks to get the best possible accuracy. We think this work would also be useful as we turn to more human involvement in training (Zaidan and Callison-Burch, 2009), as MERT in this case is intrinsically slow.

2 Unidimensional MERT

Let $\mathbf{f}_1^S = \mathbf{f}_1 \dots \mathbf{f}_S$ denote the S input sentences of our tuning set. For each sentence $\mathbf{f}_s$, let $\mathbf{C}_s = \mathbf{e}_{s,1} \dots \mathbf{e}_{s,N}$ denote a set of N candidate translations. For simplicity and without loss of generality, we assume that N is constant for each index s. Each input and output sentence pair $(\mathbf{f}_s, \mathbf{e}_{s,n})$ is weighted by a linear model that combines model parameters $\mathbf{w} = w_1 \dots w_D \in \mathbb{R}^D$ with D feature functions $h_1(\mathbf{f}, \mathbf{e}, \sim) \dots h_D(\mathbf{f}, \mathbf{e}, \sim)$, where $\sim$ is the hidden state associated with the derivation from $\mathbf{f}$ to $\mathbf{e}$, such as phrase segmentation and alignment. Furthermore, let $\mathbf{h}_{s,n} \in \mathbb{R}^D$ denote the feature vector representing the translation pair $(\mathbf{f}_s, \mathbf{e}_{s,n})$.

In MERT, the goal is to minimize an error count $E(\mathbf{r}, \mathbf{e})$ by scoring translation hypotheses against a set of reference translations $\mathbf{r}_1^S = \mathbf{r}_1 \dots \mathbf{r}_S$. Assuming as in Och (2003) that error count is additively decomposable by sentence—i.e., $E(\mathbf{r}_1^S, \mathbf{e}_1^S) = \sum_s E(\mathbf{r}_s, \mathbf{e}_s)$—this results in the following optimization problem:[2]

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \sum_{s=1}^{S} E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w})) \right\}$$
$$= \arg\min_{\mathbf{w}} \left\{ \sum_{s=1}^{S} \sum_{n=1}^{N} E(\mathbf{r}_s, \mathbf{e}_{s,n}) \delta(\mathbf{e}_{s,n}, \hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w})) \right\} \quad (1)$$

where

$$\hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w}) = \arg\max_{n \in \{1 \dots N\}} \left\{ \mathbf{w}^\intercal \mathbf{h}_{s,n} \right\}$$

The quality of this approximation is dependent on how accurately the N-best lists represent the search space of the system. Therefore, the hypothesis list is iteratively grown: decoding with an initial parameter vector seeds the N-best lists; next, parameter estimation and N-best list gathering alternate until the search space is deemed representative.

The crucial observation of Och (2003) is that the error count along any line is a piecewise constant function. Furthermore, this function for a single sentence may be computed efficiently by first finding the hypotheses that form the upper envelope of the model score function, then gathering the error count for each hypothesis along the range for which it is optimal. Error counts for the whole corpus are simply the sums of these piecewise constant functions, leading to an

[2] A metric such as TER is decomposable by sentence. BLEU is not, but its sufficient statistics are, and the literature offers several sentence-level approximations of BLEU (Lin and Och, 2004; Liang et al., 2006).

efficient algorithm for finding the global optimum of the error count along any single direction.

Such a hill-climbing algorithm in a non-convex space has no optimality guarantee: without a perfect direction finder, even a globally-exact line search may never encounter the global optimum. Coordinate ascent is often effective, though conjugate direction set finding algorithms, such as Powell's method (Powell, 1964; Press et al., 2007), or even random directions may produce better results (Cer et al., 2008). Random restarts, based on either uniform sampling or a random walk (Moore and Quirk, 2008), increase the likelihood of finding a good solution. Since random restarts and random walks lead to better solutions and faster convergence, we incorporate them into our baseline system, which we refer to as 1D-MERT.

3 Multidimensional MERT

Finding the global optimum of Eq. 1 is a difficult task, so we proceed in steps and first analyze the case where the tuning set contains only one sentence. This gives insight on how to solve the general case. With only one sentence, one of the two summations in Eq. 1 vanishes and one can exhaustively enumerate the N translations $\mathbf{e}_{1,n}$ (or $\mathbf{e}_n$ for short) to find the one that yields the minimal task loss. The only difficulty with $S = 1$ is to know for each translation $\mathbf{e}_n$ whether its feature vector $\mathbf{h}_{1,n}$ (or $\mathbf{h}_n$ for short) can be maximized using any linear model. As we can see in Fig. 1(a), some hypotheses can be maximized (e.g., $\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_4$), while others (e.g., $\mathbf{h}_3$ and $\mathbf{h}_5$) cannot. In geometric terminology, the former points are commonly called *extreme* points, and the latter are *interior* points.[3] The problem of exactly optimizing a single N-best list is closely related to the convex hull problem in computational geometry, for which generic solvers such as the QuickHull algorithm exist (Eddy, 1977; Bykat, 1978; Barber et al., 1996). A first approach would be to construct the convex hull $conv(\mathbf{h}_1 \dots \mathbf{h}_N)$ of the N-best list, then identify the point on the hull with lowest loss ($\mathbf{h}_1$ in Fig. 1) and finally compute an optimal weight vector using hull points that share common facets with the optimal feature vector ($\mathbf{h}_2$ and $\mathbf{h}_4$). Unfortunately, this doesn't quite scale even with a single N-best list, since the best known convex hull algorithm runs in $O(N^{\lfloor D/2 \rfloor + 1})$ time (Barber et al., 1996).[4]

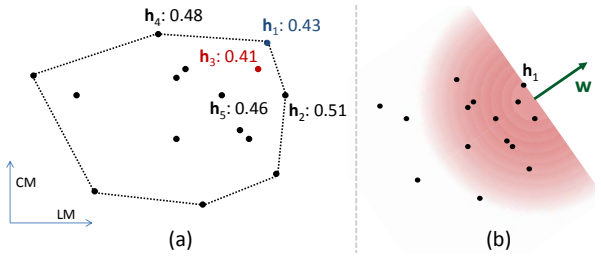


Figure 1: N-best list $(\mathbf{h}_1 \dots \mathbf{h}_N)$ with associated losses (here, TER scores) for a single input sentence, whose convex hull is displayed with dotted lines in (a). For effective visualization, our plots use only two features ($D = 2$). While we can find a weight vector that maximizes $\mathbf{h}_1$ (e.g., the $\mathbf{w}$ in (b)), no linear model can possibly maximize any of the points strictly inside the convex hull.

Algorithms presented in this paper assume that D is unrestricted, therefore we cannot afford to build any convex hull explicitly. Thus, we turn to linear programming (LP), for which we know algorithms (Karmarkar, 1984) that are polynomial in the number of dimensions and linear in the number of points, i.e., $O(NT)$, where $T = D^{3.5}$. To check if point $\mathbf{h}_i$ is extreme, we really only need to know whether we can define a half-space containing all points $\mathbf{h}_1 \dots \mathbf{h}_N$, with $\mathbf{h}_i$ lying on the hyperplane delimiting that half-space, as shown in Fig. 1(b) for $\mathbf{h}_1$. Formally, a vertex $\mathbf{h}_i$ is optimal with respect to $\arg\max_i\{\mathbf{w}^\intercal \mathbf{h}_i\}$ if and only if the following constraints hold:[5]

$$\mathbf{w}^\intercal \mathbf{h}_i = y \quad (2)$$

$$\mathbf{w}^\intercal \mathbf{h}_j \leq y, \text{ for each } j \neq i \quad (3)$$

$\mathbf{w}$ is orthogonal to the hyperplane defining the half-space, and the intercept y defines its position. The

[3] Specifically, a point $\mathbf{h}$ is extreme with respect to a convex set C (e.g., the convex hull shown in Fig. 1(a)) if it does not lie in an open line segment joining any two points of C. In a minor abuse of terminology, we sometimes simply state that a given point $\mathbf{h}$ is extreme when the nature of C is clear from context.

[4] A convex hull algorithm polynomial in D is very unlikely. Indeed, the expected number of facets of high-dimensional convex hulls grows dramatically, and—assuming a uniform distribution of points, $D = 10$, and a sufficiently large N—the expected number of facets is approximately $10^6 N$ (Buchta et al., 1985). In the worst case, the maximum number of facets of a convex hull is $O(N^{\lfloor D/2 \rfloor}/\lfloor D/2 \rfloor!)$ (Klee, 1966).

[5] A similar approach for checking whether a given point is extreme is presented in `http://www.ifor.math.ethz.ch/~fukuda/polyfaq/node22.html`, but our method generates slightly smaller LPs.

above equations represent a linear program (LP), which can be turned into canonical form

$$\begin{aligned} &\text{maximize} && \mathbf{c}^\intercal \mathbf{w} \\ &\text{subject to} && \mathbf{A}\mathbf{w} \leq \mathbf{b} \end{aligned}$$

by substituting y with $\mathbf{w}^\intercal \mathbf{h}_i$ in Eq. 3, by defining $\mathbf{A} = \{a_{n,d}\}_{1 \leq n \leq N; 1 \leq d \leq D}$ with $a_{n,d} = h_{j,d} - h_{i,d}$ (where $h_{j,d}$ is the d-th element of $\mathbf{h}_j$), and by setting $\mathbf{b} = (0, \ldots, 0)^\intercal = \mathbf{0}$. The vertex $\mathbf{h}_i$ is extreme if and only if the LP solver finds a non-zero vector $\mathbf{w}$ satisfying the canonical system. To ensure that $\mathbf{w}$ is zero only when $\mathbf{h}_i$ is interior, we set $\mathbf{c} = \mathbf{h}_i - \mathbf{h}_\mu$, where $\mathbf{h}_\mu$ is a point known to be inside the hull (e.g., the centroid of the N-best list).[6] In the remaining of this section, we use this LP formulation in function LINOPTIMIZER($\mathbf{h}_i; \mathbf{h}_1 \ldots \mathbf{h}_N$), which returns the weight vector $\hat{\mathbf{w}}$ maximizing $\mathbf{h}_i$, or which returns $\mathbf{0}$ if $\mathbf{h}_i$ is interior to $conv(\mathbf{h}_1 \ldots \mathbf{h}_N)$. We also use $conv(\mathbf{h}_i; \mathbf{h}_1 \ldots \mathbf{h}_N)$ to denote whether $\mathbf{h}_i$ is extreme with respect to this hull.

Algorithm 1: LP-MERT (for $S = 1$).

input : sent.-level feature vectors $H = \{\mathbf{h}_1 \ldots \mathbf{h}_N\}$
input : sent.-level task losses $E_1 \ldots E_N$, where $E_n := E(\mathbf{r}_1, \mathbf{e}_{1,n})$
output : optimal weight vector $\hat{\mathbf{w}}$

```
1 begin
    ▷ sort N-best list by increasing losses:
2   (i_1 ... i_N) ← IndexSort(E_1 ... E_N)
3   for n ← 1 to N do
      ▷ find ŵ maximizing i_n-th element:
4     ŵ ← LinOptimizer(h_{i_n}; H)
5     if ŵ ≠ 0 then
6       return ŵ
7   return 0
```

An exact search algorithm for optimizing a single N-best list is shown above. It lazily enumerates feature vectors in increasing order of task loss, keeping only the extreme ones. Such a vertex $\mathbf{h}_j$ is known to be on the convex hull, and the returned vector $\hat{\mathbf{w}}$ maximizes it. In Fig. 1, it would first run LINOPTIMIZER on $\mathbf{h}_3$, discard it since it is interior, and finally accept the extreme point $\mathbf{h}_1$. Each execution of LINOPTIMIZER requires $O(NT)$ time with the interior point method of (Karmarkar, 1984), and since the main loop may run $O(N)$ times in the worst case, time complexity is $O(N^2T)$. Finally, Fig. 2 empirically demonstrates the effectiveness of a linear programming approach, which in practice is seldom affected by D.

[6] We assume that $\mathbf{h}_1 \ldots \mathbf{h}_N$ are not degenerate, i.e., that they collectively span $\mathbb{R}^D$. Otherwise, all points are necessarily on the hull, yet some of them may not be uniquely maximized.

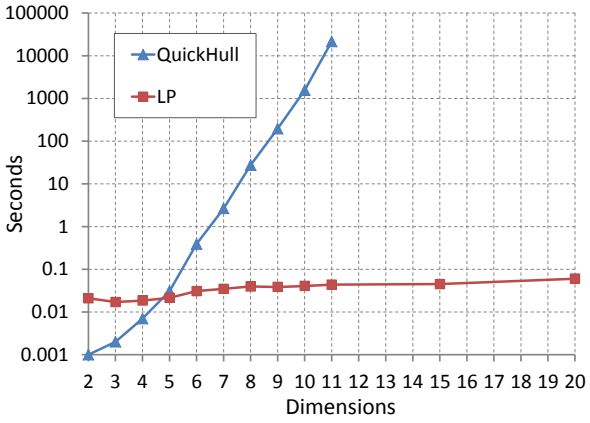


Figure 2: Running times to exactly optimize N-best lists with an increasing number of dimensions. To determine which feature vectors were on the hull, we use either linear programming (Karmarkar, 1984) or one of the most efficient convex hull computation tools (Barber et al., 1996).

3.1 Exact search: general case

We now extend LP-MERT to the general case, in which we are optimizing multiple sentences at once. This creates an intricate optimization problem, since the inner summations over $n = 1 \ldots N$ in Eq. 1 can't be optimized independently. For instance, the optimal weight vector for sentence $s = 1$ may be suboptimal with respect to sentence $s = 2$. So we need some means to determine whether a selection $\mathbf{m} = m(1) \ldots m(S) \in \mathcal{M} = [1, N]^S$ of feature vectors $\mathbf{h}_{1,m(1)} \ldots \mathbf{h}_{S,m(S)}$ is extreme, that is, whether we can find a weight vector that maximizes each $\mathbf{h}_{s,m(s)}$. Here is a reformulation of Eq. 1 that makes this condition on extremity more explicit:

$$\hat{\mathbf{m}} = \underset{\substack{conv(\mathbf{h}[\mathbf{m}]; H) \\ \mathbf{m} \in \mathcal{M}}}{\arg\min} \left\{ \sum_{s=1}^{S} E(\mathbf{r}_s, \mathbf{e}_{s,m(n)}) \right\} \qquad (4)$$

where

$$\mathbf{h}[\mathbf{m}] = \sum_{s=1}^{S} \mathbf{h}_{s,m(s)}$$

$$H = \bigcup_{\mathbf{m}' \in \mathcal{M}} \mathbf{h}[\mathbf{m}']$$

One naïve approach to address this optimization problem is to enumerate all possible combinations among the S distinct N-best lists, determine for each combination $\mathbf{m}$ whether $\mathbf{h}[\mathbf{m}]$ is extreme, and return the extreme combination with lowest total loss. It is evident that this approach is optimal (since it follows directly from Eq. 4), but it is prohibitively slow since it processes $O(N^S)$ vertices to determine whether they are extreme, which thus requires $O(N^S T)$ time per LP optimization and $O(N^{2S} T)$ time in total. We now present several improvements to make this approach more practical.

3.1.1 Sparse hypothesis combination

In the naïve approach presented above, each LP computation to evaluate $conv(\mathbf{h}[\mathbf{m}]; H)$ requires $O(N^S T)$ time since H contains N^S vertices, but we show here how to reduce it to $O(NST)$ time. This improvement exploits the fact that we can eliminate the majority of the N^S points of H, since only $S(N-1)+1$ are really needed to determine whether $\mathbf{h}[\mathbf{m}]$ is extreme. This is best illustrated using an example, as shown in Fig. 3. Both $\mathbf{h}_{1,1}$ and $\mathbf{h}_{2,1}$ in (a) and (b) are extreme with respect to their own N-best list, and we ask whether we can find a weight vector that maximizes both $\mathbf{h}_{1,1}$ and $\mathbf{h}_{2,1}$. The algorithmic trick is to geometrically translate one of the two N-best lists so that $\mathbf{h}_{1,1} = \mathbf{h}'_{2,1}$, where $\mathbf{h}'_{2,1}$ is the translation of $\mathbf{h}'_{2,1}$. Then we use linear programming with the new set of $2N-1$ points, as shown in (c), to determine whether $\mathbf{h}_{1,1}$ is on the hull, in which case the answer to the original question is yes. In the case of the combination of $\mathbf{h}_{1,1}$ and $\mathbf{h}_{2,2}$, we see in (d) that the combined set of points prevents the maximization $\mathbf{h}_{1,1}$, since this point is clearly no longer on the hull. Hence, the combination ($\mathbf{h}_{1,1}$,$\mathbf{h}_{2,2}$) cannot be maximized using any linear model. This trick generalizes to $S \geq 2$. In both (c) and (d), we used $S(N-1)+1$ points instead of N^S to determine whether a given point is extreme. We show in the appendix that this simplification does not sacrifice optimality.

3.1.2 Lazy enumeration, divide-and-conquer

Now that we can determine whether a given combination is extreme, we must next enumerate candidate combinations to find the combination that has lowest task loss among all of those that are extreme. Since the number of feature vector combinations is $O(N^S)$, exhaustive enumeration is not a reasonable

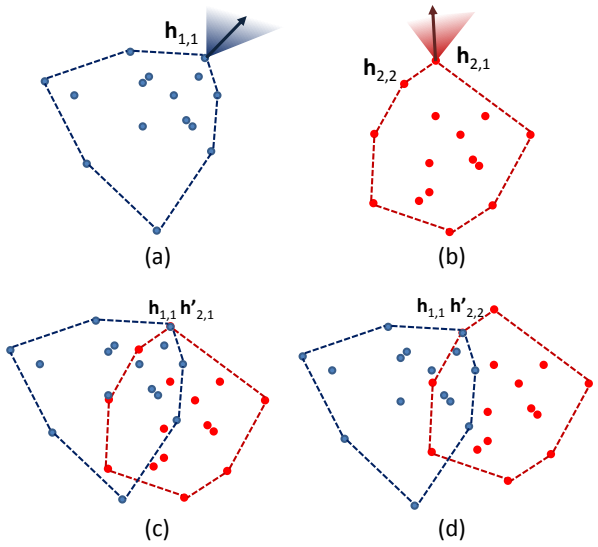


Figure 3: Given two N-best lists, (a) and (b), we use linear programming to determine which hypothesis combinations are extreme. For instance, the combination $\mathbf{h}_{1,1}$ and $\mathbf{h}_{2,1}$ is extreme (c), while $\mathbf{h}_{1,1}$ and $\mathbf{h}_{2,2}$ is not (d).

option. Instead, we use lazy enumeration to process combinations in increasing order of task loss, which ensures that the first extreme combination for $s = 1 \ldots S$ that we encounter is the optimal one. An S-ary lazy enumeration would not be particularly efficient, since the runtime is still $O(N^S)$ in the worst case. LP-MERT instead uses divide-and-conquer and binary lazy enumeration, which enables us to discard early on combinations that are not extreme. For instance, if we find that ($\mathbf{h}_{1,1}$,$\mathbf{h}_{2,2}$) is interior for sentences $s = 1, 2$, the divide-and-conquer branch for $s = 1 \ldots 4$ never actually receives this bad combination from its left child, thus avoiding the cost of enumerating combinations that are known to be interior, e.g., ($\mathbf{h}_{1,1}$,$\mathbf{h}_{2,2}$, $\mathbf{h}_{3,1}$,$\mathbf{h}_{4,1}$).

The LP-MERT algorithm for the general case is shown as Algorithm 2. It basically only calls a recursive divide-and-conquer function (GETNEXTBEST) for sentence range $1 \ldots S$. The latter function uses binary lazy enumeration in a manner similar to (Huang and Chiang, 2005), and relies on two global variables: $\mathcal{I}$ and $\mathcal{L}$. The first of these, $\mathcal{I}$, is used to memoize the results of calls to GETNEXTBEST; given a range of sentences and a rank n, it stores the nth best combination for that range of sentences. The global variable $\mathcal{L}$ stores hypotheses combination matrices, one matrix for each range of sentences (s, t) as shown in

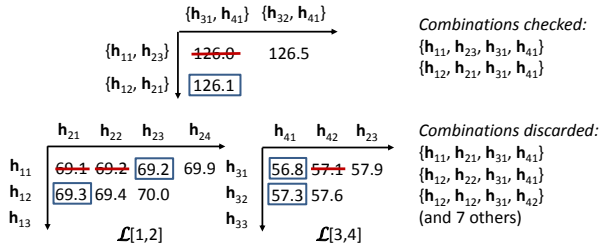


Figure 4: LP-MERT minimizes loss (TER) on four sentences. $O(N^4)$ translation combinations are possible, but the LP-MERT algorithm only tests two full combinations. Without divide-and-conquer—i.e., using 4-ary lazy enumeration—ten full combinations would have been checked unnecessarily.

Algorithm 2: LP-MERT

input : feature vectors $\mathbf{H} = \{\mathbf{h}_{s,n}\}_{1 \le s \le S; 1 \le n \le N}$
input : task losses $\mathbf{E} = \{E_{s,n}\}_{1 \le s \le S; 1 \le n \le N}$, where sent.-level costs $E_{s,n} := E(\mathbf{r}_s, \mathbf{e}_{s,n})$
output : optimal weight vector $\hat{\mathbf{w}}$ and its loss L

1 **begin**
  ▷ *sort N-best lists by increasing losses:*
2 **for** $s \leftarrow 1$ **to** S **do**
3 $(i_{s,1}..i_{s,N}) \leftarrow \text{IndexSort}(E_{s,1}..E_{s,N})$
  ▷ *find best hypothesis combination for* $1 \ldots S$*:*
4 $(\mathbf{h}_*, H_*, L) \leftarrow \text{GetNextBest}(\mathbf{H}, \mathbf{E}, 1, S)$
5 $\hat{\mathbf{w}} \leftarrow \text{LinOptimizer}(\mathbf{h}_*; H_*)$
6 **return** $(\hat{\mathbf{w}}, L)$

Fig. 4, to determine which combination to try next. The function ExpandFrontier returns the indices of unvisited cells that are adjacent (right or down) to visited cells and that might correspond to the next best hypothesis. Once no more cells need to be added to the frontier, LP-MERT identifies the lowest loss combination on the frontier (BestInFrontier), and uses LP to determine whether it is extreme. To do so, it first generates an LP using Combine, a function that implements the method described in Fig. 3. If the LP offers no solution, this combination is ignored. LP-MERT iterates until it finds a cell entry whose combination is extreme. Regarding ranges of length one ($s = t$), lines 3-10 are similar to Algorithm 1 for $S = 1$, but with one difference: GetNextBest may be called multiple times with the same argument s, since the first output of GetNextBest might not be extreme when combined with other feature vectors. Lines 3-10 of GetNextBest handle this case efficiently, since the algorithm resumes at the $(n+1)$-th

Function GetNextBest($\mathbf{H}$,$\mathbf{E}$,s,t)

input : sentence range (s, t)
output : $\mathbf{h}_*$: current best extreme vertex
output : H_*: constraint vertices
output : L: task loss of $\mathbf{h}_*$

▷ *Losses of partial hypotheses:*
1 $\mathbf{L} \leftarrow \mathcal{L}[s, t]$
2 **if** $s = t$ **then**
  ▷ *n is the index where we left off last time:*
3 $n \leftarrow \text{NbRows}(\mathbf{L})$
4 $H_s \leftarrow \{\mathbf{h}_{s,1} \ldots \mathbf{h}_{s,N}\}$
5 **repeat**
6 $n \leftarrow n + 1$
7 $\hat{\mathbf{w}} \leftarrow \text{LinOptimizer}(\mathbf{h}_{s,i_n}; H_s)$
8 $\mathbf{L}[n, 1] \leftarrow E_{s,i_n}$
9 **until** $\hat{\mathbf{w}} \neq \mathbf{0}$
10 **return** $(\mathbf{h}_{s,i_n}, H_s, \mathbf{L}[n, 1])$
11 **else**
12 $u \leftarrow \lfloor (s + t)/2 \rfloor$, $v \leftarrow u + 1$
13 **repeat**
14 **while** $\text{HasIncompleteFrontier}(\mathbf{L})$ **do**
15 $(m, n) \leftarrow \text{ExpandFrontier}(\mathbf{L})$
16 $x \leftarrow \text{NbRows}(\mathbf{L})$
17 $y \leftarrow \text{NbColumns}(\mathbf{L})$
18 **for** $m' \leftarrow x + 1$ **to** m **do**
19 $\mathcal{I}[s, u, m'] \leftarrow \text{GetNextBest}(\mathbf{H}, \mathbf{E}, s, u)$
20 **for** $n' \leftarrow y + 1$ **to** n **do**
21 $\mathcal{I}[v, t, n'] \leftarrow \text{GetNextBest}(\mathbf{H}, \mathbf{E}, v, t)$
22 $\mathbf{L}[m, n] \leftarrow \text{Loss}(\mathcal{I}[s, u, m]) + \text{Loss}(\mathcal{I}[v, t, n])$
23 $(m, n) \leftarrow \text{BestInFrontier}(\mathbf{L})$
24 $(\mathbf{h}_m, H_m, L_m) \leftarrow \mathcal{I}[s, u, m]$
25 $(\mathbf{h}_n, H_n, L_n) \leftarrow \mathcal{I}[v, t, n]$
26 $(\mathbf{h}_*, H_*) \leftarrow \text{Combine}(\mathbf{h}_m, H_m, \mathbf{h}_n, H_n)$
27 $\hat{\mathbf{w}} \leftarrow \text{LinOptimizer}(\mathbf{h}_*; H_*)$
28 **until** $\hat{\mathbf{w}} \neq \mathbf{0}$
29 **return** $(\mathbf{h}_*, H_*, \mathbf{L}[m, n])$

element of the N-best list (where n is the position where the previous execution left off).[7] We can see that a strength of this algorithm is that inconsistent combinations are deleted as soon as possible, which allows us to discard fruitless candidates *en masse*.

3.2 Approximate Search

We will see in Section 5 that our exact algorithm is often too computationally expensive in practice to be used with either a large number of sentences or a large number of features. We now present two

[7] Each N-best list is augmented with a placeholder hypothesis with loss $+\infty$. This ensures n never runs out of bounds at line 7.

Function Combine($\mathbf{h}, H, \mathbf{h}', H'$)

input : H, H': constraint vertices
input : $\mathbf{h}, \mathbf{h}'$: extreme vertices, wrt. H and H'
output : $\mathbf{h}_*, H_*$: combination as in Sec. 3.1.1
for $i \leftarrow 1$ **to** $size(H)$ **do**
  $H_i \leftarrow H_i + h'$
for $i \leftarrow 1$ **to** $size(H')$ **do**
  $H'_i \leftarrow H'_i + h$
return $(\mathbf{h} + \mathbf{h}', H \cup H')$

approaches to make LP-MERT more scalable, with the downside that we may allow search errors.

In the first case, we make the assumption that we have an initial weight vector $\mathbf{w}_0$ that is a reasonable approximation of $\hat{\mathbf{w}}$, where $\mathbf{w}_0$ may be obtained either by using a fast MERT algorithm like 1D-MERT, or by reusing the weight vector that is optimal with respect to the previous iteration of MERT. The idea then is to search only the set of weight vectors that satisfy $\cos(\hat{\mathbf{w}}, \mathbf{w}_0) \geq t$, where t is a threshold on cosine similarity provided by the user. The larger the t, the faster the search, but at the expense of more search errors. This is implemented with two simple changes in our algorithm. First, LINOPTIMIZER sets the objective vector $\mathbf{c} = \mathbf{w}_0$. Second, if the output $\hat{\mathbf{w}}$ originally returned by LINOPTIMIZER does not satisfy $\cos(\hat{\mathbf{w}}, \mathbf{w}_0) \geq t$, then it returns $\mathbf{0}$. While this modification of our algorithm may lead to search errors, it nevertheless provides some theoretical guarantee: our algorithm finds the global optimum if it lies within the region defined by $\cos(\hat{\mathbf{w}}, \mathbf{w}_0) \geq t$.

The second method is a beam approximation of LP-MERT, which normally deals with linear programs that are increasingly large in the upper branches of GETNEXTBEST's recursive calls. The main idea is to prune the output of COMBINE (line 26) by model score with respect to $\mathbf{w}_{best}$, where $\mathbf{w}_{best}$ is our current best model on the entire tuning set. Note that beam pruning can discard $\mathbf{h}_*$ (the current best extreme vertex), in which case LINOPTIMIZER returns $\mathbf{0}$. $\mathbf{w}_{best}$ is updated as follows: each time we produce a new non-zero $\hat{\mathbf{w}}$, run $\mathbf{w}_{best} \leftarrow \hat{\mathbf{w}}$ if $\hat{\mathbf{w}}$ has a lower loss than $\mathbf{w}_{best}$ on the entire tuning set. The idea of using a beam here is similar to using cosine similarity (since $\mathbf{w}_{best}$ constrains the search towards a promising region), but beam pruning also helps reduce LP optimization time and thus enables us to explore a wider space. Since $\mathbf{w}_{best}$ often improves during search, it is useful to run multiple iterations of LP-MERT until $\mathbf{w}_{best}$ doesn't change. Two or three iterations suffice in our experience. In our experiments, we use a beam size of 1000.

4 Experimental Setup

Our experiments in this paper focus on only the application of machine translation, though we believe that the current approach is agnostic to the particular system used to generate hypotheses. Both phrase-based systems (e.g., Koehn et al. (2007)) and syntax-based systems (e.g., Li et al. (2009), Quirk et al. (2005)) commonly use MERT to train free parameters. Our experiments use a syntax-directed translation approach (Quirk et al., 2005): it first applies a dependency parser to the source language data at both training and test time. Multi-word translation mappings constrained to be connected subgraphs of the source tree are extracted from the training data; these provide most lexical translations. Partially lexicalized templates capturing reordering and function word insertion and deletion are also extracted. At runtime, these mappings and templates are used to construct transduction rules to convert the source tree into a target string. The best transduction is sought using approximate search techniques (Chiang, 2007).

Each hypothesis is scored by a relatively standard set of features. The mappings contain five features: maximum-likelihood estimates of source given target and vice versa, lexical weighting estimates of source given target and vice versa, and a constant value that, when summed across a whole hypothesis, indicates the number of mappings used. For each template, we include a maximum-likelihood estimate of the target reordering given the source structure. The system may fall back to templates that mimic the source word order; the count of such templates is a feature. Likewise we include a feature to count the number of source words deleted by templates, and a feature to count the number of target words inserted by templates. The log probability of the target string according to a language models is also a feature; we add one such feature for each language model. We include the number of target words as features to balance hypothesis length.

For the present system, we use the training data of WMT 2010 to construct and evaluate an English-to-

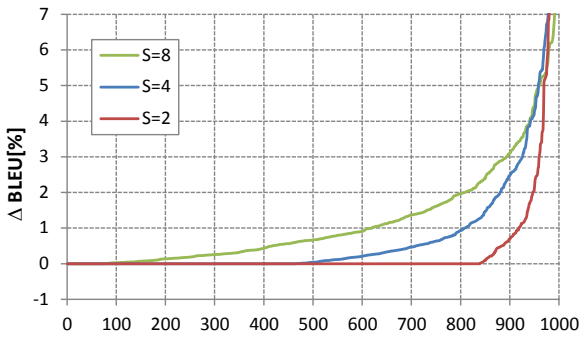


Figure 5: Line graph of sorted differences in BLEUn4r1[%] scores between LP-MERT and 1D-MERT on 1000 tuning sets of size $S = 2, 4, 8$. The highest differences for $S = 2, 4, 8$ are respectively 23.3, 19.7, 13.1.

German translation system. This consists of approximately 1.6 million parallel sentences, along with a much larger monolingual set of monolingual data. We train two language models, one on the target side of the training data (primarily parliamentary data), and the other on the provided monolingual data (primarily news). The 2009 test set is used as development data for MERT, and the 2010 one is used as test data. The resulting system has 13 distinct features.

5 Results

The section evaluates both the exact and beam version of LP-MERT. Unless mentioned otherwise, the number of features is $D = 13$ and the N-best list size is 100. Translation performance is measured with a sentence-level version of BLEU-4 (Lin and Och, 2004), using one reference translation. To enable legitimate comparisons, LP-MERT and 1D-MERT are evaluated on the *same* combined N-best lists, even though running multiple iterations of MERT with either LP-MERT or 1D-MERT would normally produce different combined N-best lists. We use WMT09 as tuning set, and WMT10 as test set. Before turning to large tuning sets, we first evaluate exact LP-MERT on data sizes that it can easily handle. Fig. 5 offers a comparison with 1D-MERT, for which we split the tuning set into 1,000 overlapping subsets for $S = 2, 4, 8$ on a combined N-best after five iterations of MERT with an average of 374 translation per sentence. The figure shows that LP-MERT never underperforms 1D-MERT in any of the 3,000 experiments, and this almost certainly confirms that

length	tested comb.	total comb.	order
8	639,960	1.33×10^{20}	$O(N^8)$
4	134,454	2.31×10^{10}	$O(2N^4)$
2	49,969	430,336	$O(4N^2)$
1	1,059	2,624	$O(8N)$

Table 1: Number of tested combinations for the experiments of Fig. 5. LP-MERT with $S = 8$ checks only 600K full combinations on average, much less than the total number of combinations (which is more than 10^{20}).

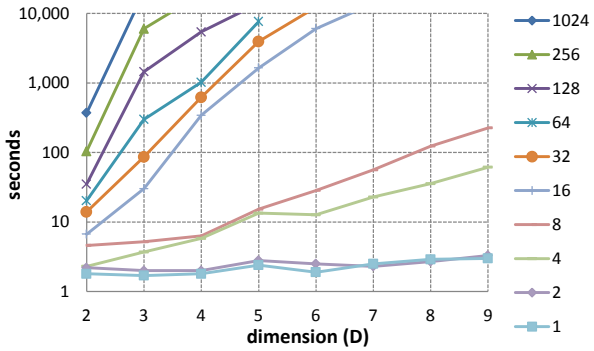


Figure 6: Effect of the number of features (runtime on 1 CPU of a modern computer). Each curve represents a different number of tuning sentences.

LP-MERT systematically finds the global optimum. In the case $S = 1$, Powell rarely makes search errors (about 15%), but the situation gets worse as S increases. For $S = 4$, it makes search errors in 90% of the cases, despite using 20 random starting points.

Some combination statistics for S up to 8 are shown in Tab. 1. The table shows the speedup provided by LP-MERT is very substantial when compared to exhaustive enumeration. Note that this is using $D = 13$, and that pruning is much more effective with less features, a fact that is confirmed in Fig. 6. $D = 13$ makes it hard to use a large tuning set, but the situation improves with $D = 2 \ldots 5$.

Fig. 7 displays execution times when LP-MERT constrains the output $\hat{\mathbf{w}}$ to satisfy $\cos(\mathbf{w_0}, \hat{\mathbf{w}}) \geq t$, where t is on the x-axis of the figure. The figure shows that we can scale to 1000 sentences when (exactly) searching within the region defined by $\cos(\mathbf{w_0}, \hat{\mathbf{w}}) \geq .84$. All these running times would improve using parallel computing, since divide-and-conquer algorithms are generally easy to parallelize.

We also evaluate the beam version of LP-MERT, which allows us to exploit tuning sets of reasonable

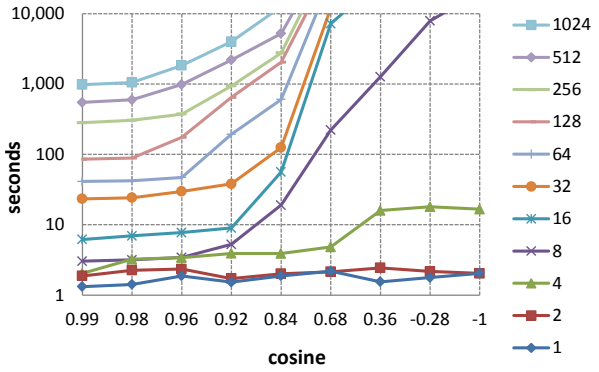


Figure 7: Effect of a constraint on $\mathbf{w}$ (runtime on 1 CPU).

	32	64	128	256	512	1024
1D-MERT	22.93	20.70	18.57	16.07	15.00	15.44
our work	25.25	22.28	19.86	17.05	15.56	15.67
	+2.32	+1.59	+1.29	+0.98	+0.56	+0.23

Table 2: BLEUn4r1[%] scores for English-German on WMT09 for tuning sets ranging from 32 to 1024 sentences.

size. Results are displayed in Table 2. The gains are fairly substantial, with gains of 0.5 BLEU point or more in all cases where $S \leq 512$.[8] Finally, we perform an end-to-end MERT comparison, where both our algorithm and 1D-MERT are iteratively used to generate weights that in turn yield new N-best lists. Tuning on 1024 sentences of WMT10, LP-MERT converges after seven iterations, with a BLEU score of 16.21%; 1D-MERT converges after nine iterations, with a BLEU score of 15.97%. Test set performance on the full WMT10 test set for LP-MERT and 1D-MERT are respectively 17.08% and 16.91%.

6 Related Work

One-dimensional MERT has been very influential. It is now used in a broad range of systems, and has been improved in a number of ways. For instance, lattices or hypergraphs may be used in place of N-best lists to form a more comprehensive view of the search space with fewer decoding runs (Macherey et al., 2008; Kumar et al., 2009; Chatterjee and Cancedda, 2010). This particular refinement is orthogonal to our approach, though. We expect to extend LP-MERT to hypergraphs in future work. Exact search may be challenging due to the computational complexity of the search space (Leusch et al., 2008), but approximate search should be feasible.

Other research has explored alternate methods of gradient-free optimization, such as the downhill-simplex algorithm (Nelder and Mead, 1965; Zens et al., 2007; Zhao and Chen, 2009). Although the search space is different than that of Och's algorithm, it still relies on one-dimensional line searches to reflect, expand, or contract the simplex. Therefore, it suffers the same problems of one-dimensional MERT: feature sets with complex non-linear interactions are difficult to optimize. LP-MERT improves on these methods by searching over a larger subspace of parameter combinations, not just those on a single line.

We can also change the objective function in a number of ways to make it more amenable to optimization, leveraging knowledge from elsewhere in the machine learning community. Instance reweighting as in boosting may lead to better parameter inference (Duh and Kirchhoff, 2008). Smoothing the objective function may allow differentiation and standard ML learning techniques (Och and Ney, 2002). Smith and Eisner (2006) use a smoothed objective along with deterministic annealing in hopes of finding good directions and climbing past locally optimal points. Other papers use margin methods such as MIRA (Watanabe et al., 2007; Chiang et al., 2008), updated somewhat to match the MT domain, to perform incremental training of potentially large numbers of features. However, in each of these cases the objective function used for training no longer matches the final evaluation metric.

7 Conclusions

Our primary contribution is the first known exact search algorithm for direct loss minimization on N-best lists in multiple dimensions. Additionally, we present approximations that consistently outperform standard one-dimensional MERT on a competitive machine translation system. While Och's method of MERT is generally quite successful, there are cases where it does quite poorly. A more global search such as LP-MERT lowers the expected risk of such poor solutions. This is especially important for current machine translation systems that rely heavily on MERT, but may also be valuable for other textual ap-

[8] One interesting observation is that the performance of 1D-MERT degrades as S grows from 2 to 8 (Fig. 5), which contrasts with the results shown in Tab. 2. This may have to do with the fact that N-best lists with $S = 2$ have much fewer local maxima than with $S = 4, 8$, in which case 20 restarts is generally enough.

plications. Recent speech recognition systems have also explored combinations of more acoustic and language models, with discriminative training of 5-10 features rather than one million (Lööf et al., 2010); LP-MERT could be valuable here as well.

The one-dimensional algorithm of Och (2003) has been subject to study and refinement for nearly a decade, while this is the first study of multi-dimensional approaches. We demonstrate the potential of multi-dimensional approaches, but we believe there is much room for improvement in both scalability and speed. Furthermore, a natural line of research would be to extend LP-MERT to compact representations of the search space, such as hypergraphs.

There are a number of broader implications from this research. For instance, LP-MERT can aid in the evaluation of research on MERT. This approach supplies a truly optimal vector as ground truth, albeit under limited conditions such as a constrained direction set, a reduced number of features, or a smaller set of sentences. Methods can be evaluated based on not only improvements over prior approaches, but also based on progress toward a global optimum.

Acknowledgements

We thank Xiaodong He, Kristina Toutanova, and three anonymous reviewers for their valuable suggestions.

Appendix A: Proof of optimality

In this appendix, we prove that LP-MERT (Algorithm 2) is exact. As noted before, the naïve approach of solving Eq. 4 is to enumerate all $O(N^S)$ hypotheses combinations in $\mathcal{M}$, discard the ones that are not extreme, and return the best scoring one. LP-MERT relies on algorithmic improvements to speed up this approach, and we now show that none of them affect the optimality of the solution.

Divide-and-conquer. Divide-and-conquer in Algorithm 2 discards any partial hypothesis combination $\mathbf{h}[m(j)\dots m(k)]$ if it is not extreme, even before considering any extension $\mathbf{h}[m(i)\dots m(j)\dots m(k)\dots m(l)]$. This does not sacrifice optimality, since if $conv(\mathbf{h}; H)$ is false, then $conv(\mathbf{h}; H \cup G)$ is false for any set G.

Proof: Assume $conv(\mathbf{h}; H)$ is false, so $\mathbf{h}$ is interior to H. By definition, any interior point $\mathbf{h}$ can be written as a linear combination of other points: $\mathbf{h} = \sum_i \lambda_i \mathbf{h}_i$, with $\forall i(\mathbf{h}_i \in H, \mathbf{h}_i \neq \mathbf{h}, \lambda_i \geq 0)$ and $\sum_i \lambda_i = 1$. This same combination of points also demonstrates that $\mathbf{h}$ is interior to $H \cup G$, thus $conv(\mathbf{h}; H \cup G)$ is false as well.

Sparse hypothesis combination. We show here that the simplification of linear programs in Section 3.1.1 from size $O(N^S)$ to size $O(NS)$ does not change the value of $conv(\mathbf{h}; H)$. More specifically, this means that linear optimization of the output of the COMBINE method at lines 26-27 of function GETNEXTBEST does not introduce any error. Let $(\mathbf{g}_1 \dots \mathbf{g}_U)$ and $(\mathbf{h}_1 \dots \mathbf{h}_V)$ be two N-best lists to be combined, then:

$$conv\Big(\mathbf{g}_u + \mathbf{h}_v; \bigcup_{i=1}^{U}(\mathbf{g}_i + \mathbf{h}_v) \cup \bigcup_{j=1}^{V}(\mathbf{g}_u + \mathbf{h}_j)\Big) = conv\Big(\mathbf{g}_u + \mathbf{h}_v; \bigcup_{i=1}^{U}\bigcup_{j=1}^{V}(\mathbf{g}_i + \mathbf{h}_j)\Big)$$

Proof: To prove this equality, it suffices to show that: (1) if $\mathbf{g}_u + \mathbf{h}_v$ is interior wrt. the first $conv$ binary predicate in the above equation, then it is interior wrt. the second $conv$, and (2) if $\mathbf{g}_u + \mathbf{h}_v$ is interior wrt. the second $conv$, then it is interior wrt. the first $conv$. Claim (1) is evident, since the set of points in the first $conv$ is a subset of the other set of points. Thus, we only need to prove (2). We first geometrically translate all points by $-\mathbf{g}_u - \mathbf{h}_v$. Since $\mathbf{g}_u + \mathbf{h}_v$ is interior wrt. the second $conv$, we can write:

$$\begin{aligned}
\mathbf{0} &= \sum_{i=1}^{U}\sum_{j=1}^{V}\lambda_{i,j}(\mathbf{g}_i + \mathbf{h}_j - \mathbf{g}_u - \mathbf{h}_v) \\
&= \sum_{i=1}^{U}\sum_{j=1}^{V}\lambda_{i,j}(\mathbf{g}_i - \mathbf{g}_u) + \sum_{i=1}^{U}\sum_{j=1}^{V}\lambda_{i,j}(\mathbf{h}_j - \mathbf{h}_v) \\
&= \sum_{i=1}^{U}(\mathbf{g}_i - \mathbf{g}_u)\sum_{j=1}^{V}\lambda_{i,j} + \sum_{j=1}^{V}(\mathbf{h}_j - \mathbf{h}_v)\sum_{i=1}^{U}\lambda_{i,j} \\
&= \sum_{i=1}^{U}\lambda'_i(\mathbf{g}_i - \mathbf{g}_u) + \sum_{j=1}^{V}\lambda'_{U+j}(\mathbf{h}_j - \mathbf{h}_v)
\end{aligned}$$

where $\{\lambda'_i\}_{1\leq i\leq U+V}$ values are computed from $\{\lambda_{i,j}\}_{1\leq i\leq U, 1\leq j\leq V}$ as follows: $\lambda'_i = \sum_j \lambda_{i,j}, i \in [1, U]$ and $\lambda'_{U+j} = \sum_i \lambda_{i,j}, j \in [1, V]$. Since the interior point is $\mathbf{0}$, λ'_i values can be scaled so that they sum to 1 (necessary condition in the definition of interior points), which proves that the following predicate is false:

$$conv\Big(\mathbf{0}; \bigcup_{i=1}^{U}(\mathbf{g}_i - \mathbf{g}_u) \cup \bigcup_{j=1}^{V}(\mathbf{h}_j - \mathbf{h}_v)\Big)$$

which is equivalent to stating that the following is false:

$$conv\Big(\mathbf{g}_u + \mathbf{h}_v; \bigcup_{i=1}^{U}(\mathbf{g}_i + \mathbf{h}_v) \cup \bigcup_{j=1}^{V}(\mathbf{g}_u + \mathbf{h}_j)\Big)$$

References

C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. 1996. The QuickHull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22:469–483.

C. Buchta, J. Muller, and R. F. Tichy. 1985. Stochastical approximation of convex bodies. *Math. Ann.*, 271:225–235.

A. Bykat. 1978. Convex hull of a finite set of points in two dimensions. *Inf. Process. Lett.*, 7(6):296–298.

Daniel Cer, Dan Jurafsky, and Christopher D. Manning. 2008. Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 26–34.

Samidh Chatterjee and Nicola Cancedda. 2010. Minimum error rate training by sampling the translation lattice. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. Association for Computational Linguistics.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *EMNLP*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

W. Chou, C. H. Lee, and B. H. Juang. 1993. Minimum error rate training based on N-best string models. In *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP '93)*, pages 652–655, Vol. 2.

Kevin Duh and Katrin Kirchhoff. 2008. Beyond log-linear models: boosted minimum error rate training for programming N-best re-ranking. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 37–40, Stroudsburg, PA, USA.

William F. Eddy. 1977. A new convex hull algorithm for planar sets. *ACM Trans. Math. Softw.*, 3:398–403.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64, Stroudsburg, PA, USA.

Biing-Hwang Juang, Wu Hou, and Chin-Hui Lee. 1997. Minimum classification error rate methods for speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 5(3):257–265.

N. Karmarkar. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395.

Victor Klee. 1966. Convex polytopes and linear programming. In *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Demonstration Session*.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 163–171.

Gregor Leusch, Evgeny Matusov, and Hermann Ney. 2008. Complexity of finding the BLEU-optimal hypothesis in a confusion network. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 839–847, Stroudsburg, PA, USA.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: an open source toolkit for parsing-based MT. In *Proc. of WMT*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, Stroudsburg, PA, USA.

Jonas Lööf, Ralf Schlüter, and Hermann Ney. 2010. Discriminative adaptation for log-linear acoustic models. In *INTERSPEECH*, pages 1648–1651.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734.

David McAllester, Tamir Hazan, and Joseph Keshet. 2010. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems 23*, pages 1594–1602.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proceedings of EACL*, pages 297–304.

Robert C. Moore and Chris Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proceedings of the 22nd International*

Conference on Computational Linguistics - Volume 1, pages 585–592.

J. A. Nelder and R. Mead. 1965. A simplex method for function minimization. *Computer Journal*, 7:308–313.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proc. of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

M.J.D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.*, 7(2):155–162.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: syntactically informed phrasal SMT. In *Proc. of ACL*, pages 271–279.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 787–794, Stroudsburg, PA, USA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*, pages 223–231.

Andreas Stolcke, Yochai Knig, and Mitchel Weintraub. 1997. Explicit word error minimization in N-best list rescoring. In *In Proc. Eurospeech*, pages 163–166.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *EMNLP-CoNLL*.

Omar F. Zaidan and Chris Callison-Burch. 2009. Feasibility of human-in-the-loop minimum error rate training. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, pages 52–61.

Richard Zens, Sasa Hasan, and Hermann Ney. 2007. A systematic comparison of training criteria for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 524–532, Prague, Czech Republic.

Bing Zhao and Shengyuan Chen. 2009. A simplex Armijo downhill algorithm for optimizing statistical machine translation decoding parameters. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 21–24.

Unsupervised Structure Prediction with Non-Parallel Multilingual Guidance

Shay B. Cohen **Dipanjan Das** **Noah A. Smith**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{scohen,dipanjan,nasmith}@cs.cmu.edu

Abstract

We describe a method for prediction of linguistic structure in a language for which only unlabeled data is available, using annotated data from a set of one or more helper languages. Our approach is based on a model that locally mixes between supervised models from the helper languages. Parallel data is not used, allowing the technique to be applied even in domains where human-translated texts are unavailable. We obtain state-of-the-art performance for two tasks of structure prediction: unsupervised part-of-speech tagging and unsupervised dependency parsing.

1 Introduction

A major focus of recent NLP research has involved unsupervised learning of structure such as POS tag sequences and parse trees (Klein and Manning, 2004; Johnson et al., 2007; Berg-Kirkpatrick et al., 2010; Cohen and Smith, 2010, *inter alia*). In its purest form, such research has improved our understanding of unsupervised learning practically and formally, and has led to a wide range of new algorithmic ideas. Another strain of research has sought to exploit resources and tools in some languages (especially English) to construct similar resources and tools for other languages, through heuristic "projection" (Yarowsky and Ngai, 2001; Xi and Hwa, 2005) or constraints in learning (Burkett and Klein, 2008; Smith and Eisner, 2009; Das and Petrov, 2011; McDonald et al., 2011) or inference (Smith and Smith, 2004). Joint unsupervised learning (Snyder and Barzilay, 2008; Naseem et al., 2009; Snyder et al., 2009) is yet another research direction that seeks to learn models for many languages at once, exploiting linguistic universals and language similarity. The driving force behind all of this work has been the hope of building NLP tools for languages that lack annotated resources.[1]

In this paper, we present an approach to using annotated data from one or more languages (*helper* languages) to learn models for another language that lacks annotated data (the *target* language). Unlike the previous work mentioned above, our framework does not rely on parallel data in any form. This is advantageous because parallel text exists only in a few text domains (e.g., religious texts, parliamentary proceedings, and news).

We focus on generative probabilistic models parameterized by multinomial distributions. We begin with supervised maximum likelihood estimates for models of the helper languages. In the second stage, we learn a model for the target language using unannotated data, maximizing likelihood over *interpolations* of the helper language models' distributions. The tying is performed at the parameter level, through coarse, nearly-universal syntactic categories (POS tags). The resulting model is then used to *initialize* learning of the target language's model using standard unsupervised parameter estimation.

Some previous multilingual research, such as Bayesian parameter tying across languages (Cohen and Smith, 2009) or models of parameter

[1] Although the stated objective is often to build systems for resource-poor languages and domains, for evaluation purposes, annotated treebank test data figure prominently in this research (including in this paper).

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 50–61,
Edinburgh, Scotland, UK, July 27–31, 2011.

drift down phylogenetic trees (Berg-Kirkpatrick and Klein, 2010) is comparable, but the practical assumption of supervised helper languages is new to this work. Naseem et al. (2010) used universal syntactic categories and rules to improve grammar induction, but their model required expert handwritten rules as constraints.

Herein, we specifically focus on two problems in linguistic structure prediction: unsupervised POS tagging and unsupervised dependency grammar induction. Our experiments demonstrate that the presented method outperforms strong state-of-the-art unsupervised baselines for both tasks. Our approach can be applied to other problems in which a subset of the model parameters can be linked across languages. We also experiment with unsupervised learning of dependency structures from words, by combining our tagger and parser. Our results show that combining our tagger and parser with joint inference outperforms pipeline inference, and, in several cases, even outperforms models built using gold-standard part-of-speech tags.

2 Overview

For each language ℓ, we assume the presence of a set of fine-grained POS tags $\mathcal{F}_\ell$, used to annotate the language's treebank. Furthermore, we assume that there is a set of universal, coarse-grained POS tags $\mathcal{C}$ such that, for every language ℓ, there is a deterministic mapping from fine-grained to coarse-grained tags, $\lambda_\ell : \mathcal{F}_\ell \rightarrow \mathcal{C}$. Our approach can be summarized using the following steps for a given task:

1. Select a set of L helper languages for which there exists annotated data $\langle \mathcal{D}_1, \ldots, \mathcal{D}_L \rangle$. Here, we use treebanks in these languages.
2. For all $\ell \in \{1, \ldots, L\}$, convert the examples in $\mathcal{D}_\ell$ by applying λ_ℓ to every POS tag in the data, resulting in $\tilde{\mathcal{D}}_\ell$. Estimate the parameters of a probabilistic model using $\tilde{\mathcal{D}}_\ell$. In this work, such models are generative probabilistic models based on multinomial distributions,[2] including an HMM and the dependency model with valence (DMV) of Klein and Manning (2004). Denote the subset of parameters that are unlexicalized by $\boldsymbol{\theta}^{(\ell)}$. (Lexicalized parameters will be denoted $\boldsymbol{\eta}^{(\ell)}$.)
3. For the target language, define the set of valid unlexicalized parameters

$$\Theta = \left\{ \boldsymbol{\theta} \;\middle|\; \boldsymbol{\theta}_k = \sum_{\ell=1}^{L} \beta_{\ell,k} \boldsymbol{\theta}_k^{(\ell)}, \sum_{\ell=1}^{L} \beta_{\ell,k} = 1, \boldsymbol{\beta} \geq \mathbf{0} \right\}, \tag{1}$$

   for each group of parameters k, and maximize likelihood over that set, using the target-language unannotated data $\mathcal{U}$. Because the syntactic categories referenced by each $\boldsymbol{\theta}^{(\ell)}$ and all models in Θ are in $\mathcal{C}$, the models will be in the same parametric family. (Figure 1 gives a graphical interpretation of Θ.) Let the resulting model be $\boldsymbol{\theta}$.
4. Transform $\boldsymbol{\theta}$ by expanding the coarse-grained syntactic categories into the target language's fine-grained categories. Use the resulting model to initialize parameter estimation, this time over fine-grained tags, again using the unannotated target-language data $\mathcal{U}$. Initialize lexicalized parameters $\boldsymbol{\eta}$ for the target language using standard methods (e.g., uniform initialization with random symmetry breaking).

The main idea in the approach is to estimate a certain model family for one language, while using supervised models from other languages. The link between the languages is achieved through coarse-grained categories, which are now now commonplace (and arguably central to any theory of natural language syntax). A key novel contribution is the use of helper languages for initialization, and of unsupervised learning to learn the contribution of each helper language to that initialization (step 3). Additional treatment is required in expanding the coarse-grained model to the fine-grained one (step 4).

3 Interpolated Multilingual Probabilistic Context-Free Grammars

Our focus in this paper is on models that consist of multinomial distributions that have relationships between them through a generative process such as a probabilistic context-free grammar (PCFG). More specifically, we assume that we have a model defining a probability distribution over observed surface forms $\boldsymbol{x}$ and derivations $\boldsymbol{y}$ parametrized by $\boldsymbol{\theta}$:

[2] In §4 we also consider a feature-based parametrization.

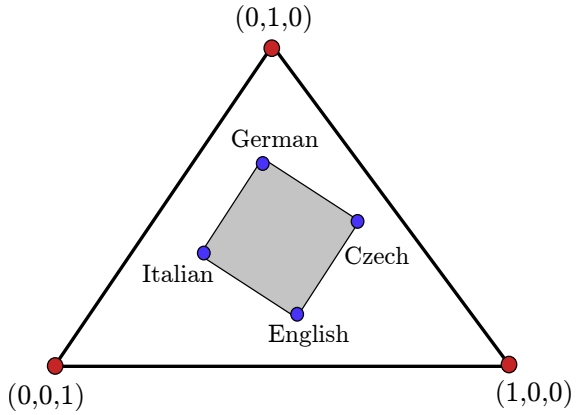


Figure 1: A simple case of interpolation within the 3-event probability simplex. The shaded area corresponds to a convex hull inside the probability simplex, indicating a mixture of the parameters of the four languages shown in the figure.

$$p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_{k=1}^{K} \prod_{i=1}^{N_k} \theta_{k,i}^{f_{k,i}(\boldsymbol{x},\boldsymbol{y})} \quad (2)$$

$$= \exp \sum_{k=1}^{K} \sum_{i=1}^{N_k} f_{k,i}(\boldsymbol{x}, \boldsymbol{y}) \log \theta_{k,i} \quad (3)$$

where $f_{k,i}$ is a function that "counts" the number of times the kth distribution's ith event occurs in the derivation. The parameters $\boldsymbol{\theta}$ are a collection of K multinomials $\langle \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K \rangle$, the kth of which includes N_k events. Letting $\boldsymbol{\theta}_k = \langle \theta_{k,1}, \ldots, \theta_{k,N_k} \rangle$, each $\theta_{k,i}$ is a probability, such that $\forall k, \forall i, \theta_{k,i} \geq 0$ and $\forall k, \sum_{i=1}^{N_k} \theta_{k,i} = 1$.

3.1 Multilingual Interpolation

Our framework places additional, temporary constraints on the parameters $\boldsymbol{\theta}$. More specifically, we assume that we have L existing, parameter estimates for the multinomial families from Eq. 3. Each such estimate $\boldsymbol{\theta}^{(\ell)}$, for $1 \leq \ell \leq L$, corresponds to a the maximum likelihood estimate based on annotated data for the ℓth helper language. Then, to create a model for new language, we define a new set of parameters $\boldsymbol{\theta}$ as:

$$\theta_{k,i} = \sum_{\ell=1}^{L} \beta_{\ell,k} \theta_{k,i}^{(\ell)}, \quad (4)$$

where $\boldsymbol{\beta}$ is the set of coefficients that we will now be interested in estimating (instead of directly estimating $\boldsymbol{\theta}$). Note that for each k, $\sum_{\ell=1}^{L} \beta_{\ell,k} = 1$ and $\beta_{\ell,k} \geq 0$.

3.2 Grammatical Interpretation

We now give an interpretation of our approach relating it to PCFGs. We assume familiarity with PCFGs. For a PCFG $\langle \boldsymbol{G}, \boldsymbol{\theta} \rangle$ we denote the set of nonterminal symbols by $\mathcal{N}$, the set of terminal symbols by Σ, and the set of rewrite rules for each nonterminal $A \in \mathcal{N}$ by $R(A)$. Each $r \in R(A)$ has the form $A \rightarrow \alpha$ where $\alpha \in (\mathcal{N} \cup \Sigma)^*$. In addition, there is a probability attached to each rule $\theta_{A \rightarrow \alpha}$ such that $\forall A \in \mathcal{N}, \sum_{\alpha:(A \rightarrow \alpha) \in R(A)} \theta_{A \rightarrow \alpha} = 1$. A PCFG can be framed as a model using Eq. 3, where $\boldsymbol{\theta}$ correspond to $K = |\mathcal{N}|$ multinomial distributions, where each distribution attaches probabilities to rules with a specific left hand symbol.

We assume that the model we are trying to estimate (over coarse part-of-speech tags) can be framed as a PCFG $\langle \boldsymbol{G}, \boldsymbol{\theta} \rangle$. This is indeed the case for part-of-speech tagging and dependency grammar induction we experiment with in §6. In that case, our approach can be framed for PCFGs as following. We assume that there exists L set of parameters for this PCFG $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(L)}$, each corresponding to a helper language. We then create a new PCFG $\boldsymbol{G}'$ with parameters $\boldsymbol{\theta}'$ and $\boldsymbol{\beta}$ as follows:

1. $\boldsymbol{G}'$ contains all nonterminal and terminal symbols in $\boldsymbol{G}$, and none of the rules in $\boldsymbol{G}$.
2. For each nonterminal A in $\boldsymbol{G}$, we create a new nonterminal $a_{A,\ell}$ for $\ell \in \{1, \ldots, L\}$.
3. For each nonterminal A in $\boldsymbol{G}$, we create rules $A \rightarrow a_{A,\ell}$ for $\ell \in \{1, \ldots, L\}$ which have probabilities $\beta_{A \rightarrow a_{A,\ell}}$.
4. For each rule $A \rightarrow \alpha$ in $\boldsymbol{G}$, we add to $\boldsymbol{G}'$ the rule $a_{A,\ell} \rightarrow \alpha$ with

$$\theta'_{a_{A,\ell} \rightarrow \alpha} = \theta^{(\ell)}_{A \rightarrow \alpha}. \quad (5)$$

where $\theta^{(\ell)}_{A \rightarrow \alpha}$ is the probability associated with rule $A \rightarrow \alpha$ in the ℓth helper language.

At each point, the derivational process of this PCFG uses the nonterminal's specific $\boldsymbol{\beta}$ coefficients

to choose one of the helper languages. It then selects a rule according to the multinomial from that language. This step is repeated until a whole derivation is generated.

This PCFG representation of the approach in §3 points to a possible generalization. Instead of using an identical CFG backbone for each language, we can use a set of PCFGs, $\langle \boldsymbol{G}^{(\ell)}, \boldsymbol{\theta}^{(\ell)} \rangle$ with an identical nonterminal set and alphabet, and repeat the same construction as above, replacing step 4 with the addition of rules of the form $a_{A,\ell} \rightarrow \alpha$ for each rule $A \rightarrow \alpha$ in $\boldsymbol{G}^{(\ell)}$. Such a construction allows more syntactic variability in the language we are trying to estimate, originating in the syntax of the various helper languages. In this paper, we do not use this generalization, and always use the same PCFG backbone for all languages.

Note that the interpolated model can still be understood in terms of the exponential model of Eq. 3. For a given collection of multinomials and base models of the form of Eq. 3, we can analogously define a new log-linear model over a set of extended derivations. These derivations will now include $L \times K$ features of the form $g_{\ell,k}(\boldsymbol{x}, \boldsymbol{y})$, corresponding to a count of the event of choosing the ℓth mixture component for multinomial k. In addition, the feature set $f_{k,i}(\boldsymbol{x}, \boldsymbol{y})$ will be extended to a feature set of the form $f_{\ell,k,i}(\boldsymbol{x}, \boldsymbol{y})$, analogous to step 4 in constructed PCFG above. The model parameterized according to Eq. 4 can be recovered by marginalizing out the "g" features. We will refer to the model with these new set of features as "the extended model."

4 Inference and Parameter Estimation

The main building block commonly required for unsupervised learning in NLP is that of computing feature expectations for a given model. These feature expectations can be used with an algorithm such as expectation-maximization (where the expectations are normalized to obtain a new set of multinomial weights) or with other gradient based log-likelihood optimization algorithms such as L-BFGS (Liu and Nocedal, 1989) for feature-rich models.

Estimating Multinomial Distributions Given a surface form $\boldsymbol{x}$, a multinomial k and an event i in the multinomial, "feature expectation" refers to the calculation of the following quantities (in the extended model):

$$\mathbb{E}[f_{\ell,k,i}(\boldsymbol{x}, \boldsymbol{y})] = \sum_{\boldsymbol{y}} p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{\theta}) f_{\ell,k,i}(\boldsymbol{x}, \boldsymbol{y}) \quad (6)$$

$$\mathbb{E}[g_{\ell,k}(\boldsymbol{x}, \boldsymbol{y})] = \sum_{\boldsymbol{y}} p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{\theta}) g_{\ell,k}(\boldsymbol{x}, \boldsymbol{y}) \quad (7)$$

These feature expectations can usually be computed using algorithms such as the forward-backward algorithm for hidden Markov models, or more generally, the inside-outside algorithm for PCFGs. In this paper, however, the task of estimation is different than the traditional task. As mentioned in §2, we are interested in estimating $\boldsymbol{\beta}$ from Eq. 4, while fixing $\boldsymbol{\theta}^{(\ell)}$. Therefore, we are only interested in computing expectations of the form of Eq. 7.

As explained in §3.2, any model interpolating with the $\boldsymbol{\beta}$ parameters can be reduced to a new log-linear model with additional features representing the mixture coefficients of $\boldsymbol{\beta}$. We can then use the inside-outside algorithm to obtain the necessary feature expectations for features of the form $g_{\ell,k}(\boldsymbol{x}, \boldsymbol{y})$, expectations which assist in the estimation of the $\boldsymbol{\beta}$ parameters.

These feature expectations can readily be used in estimation algorithms such as expectation-maximization (EM). With EM, the update at iteration t would be:

$$\beta_{\ell,k}^{(t)} = \frac{\mathbb{E}[g_{\ell,k}(\boldsymbol{x}, \boldsymbol{y})]}{\sum_{\ell} \mathbb{E}[g_{\ell,k}(\boldsymbol{x}, \boldsymbol{y})]}, \quad (8)$$

where the expectations are taken with respect to $\boldsymbol{\beta}^{(t-1)}$ and the fixed $\boldsymbol{\theta}^{(l)}$ for $\ell = 1, \ldots, L$.

Estimating Feature-Rich Directed Models Recently Berg-Kirkpatrick et al. (2010) found that replacing traditional multinomial parameterizations with locally normalized, feature-based log-linear models was advantageous. This can be understood as parameterizing $\boldsymbol{\theta}$:

$$\theta_{k,i} = \frac{\exp \boldsymbol{\psi}^{\top} \mathbf{h}(k, i)}{\sum_{i'} \exp \boldsymbol{\psi}^{\top} \mathbf{h}(k, i')} \quad (9)$$

where $\mathbf{h}(k, i)$ are a set of features looking at event i in context k. For such a feature-rich model, our multilingual modeling framework still substitutes $\boldsymbol{\theta}$ with a mixture of supervised multinomials for L helper languages as in Eq. 4. However, for computational

convenience, we also reparametrize the mixture coefficients $\boldsymbol{\beta}$:

$$\beta_{\ell,k} = \frac{\exp \gamma_{\ell,k}}{\sum_{\ell'=1}^{L} \exp \gamma_{\ell',k}} \quad (10)$$

Here, each $\gamma_{\ell,k}$ is an unconstrained parameter, and the above "softmax" transformation ensures that $\boldsymbol{\beta}$ lies within the probability simplex for context k. This is done so that a gradient-based optimization method like L-BFGS (Liu and Nocedal, 1989) can be used to estimate $\boldsymbol{\gamma}$ without having to worry about additional simplex constraints. For optimization, derivatives of the data log-likelihood with respect to $\boldsymbol{\gamma}$ need to be computed. We calculate the derivatives following Berg-Kirkpatrick et al. (2010, §3.1), making use of feature expectations, calculated exactly as before.

In addition to these estimation techniques, which are based on the optimization of the log-likelihood, we also consider a trivially simple technique for estimating $\boldsymbol{\beta}$: setting $\beta_{l,k}$ to the uniform weight L^{-1}, where L is the number of helper languages.

5 Coarse-to-Fine Multinomial Expansion

To expand these multinomials involving coarse-grained categories into multinomials over fine-grained categories specific to the target language t, we do the following:

- Whenever a multinomial *conditions* on a coarse category $c \in \mathcal{C}$, we make copies of it for each fine-grained category in $\lambda_t^{-1}(c) \subset \mathcal{F}_t$.[3] If the multinomial does not condition on coarse categories, it is simply copied.
- Whenever a probability θ_i within a multinomial distribution involves a coarse-grained category c as an event (i.e., it is on the left side of the conditional bar), we expand the event into $|\lambda_t^{-1}(c)|$ new events, one per corresponding fine-grained category, each assigned the value $\frac{\theta_i}{|\lambda_t^{-1}(c)|}$.[4]

[3] We note that in the models we experiment with, we always condition on at most one fine-grained category.

[4] During this expansion process for a coarse event, we tried adding random noise to $\frac{\theta_i}{|\lambda_t^{-1}(c)|}$ and renormalizing, to break symmetry between the fine events, but that was found to be harmful in preliminary experiments.

The result of this expansion is a model in the desired family; we use it to initialize conventional unsupervised parameter estimation. Lexical parameters, if any, do not undergo this expansion process, and they are estimated anew in the fine grained model during unsupervised learning, and are initialized using standard methods.

6 Experiments and Results

In this section, we describe the experiments undertaken and the results achieved. We first note the characteristics of the datasets and the universal POS tags used in multilingual modeling.

6.1 Data

For our experiments, we fixed a set of four helper languages with relatively large amounts of data, displaying nontrivial linguistic diversity: Czech (Slavic), English (West-Germanic), German (West-Germanic), and Italian (Romance). The datasets are the CoNLL-X shared task data for Czech and German (Buchholz and Marsi, 2006),[5] the Penn Treebank for English (Marcus et al., 1993), and the CoNLL 2007 shared task data for Italian (Montemagni et al., 2003). This was the only set of helper languages we tested; improvements are likely possible. We leave an exploration of helper language choice (a subset selection problem) to future research, instead demonstrating that the concept has merit.

We considered ten target languages: Bulgarian (Bg), Danish (Da), Dutch (Nl), Greek (El), Japanese (Jp), Portuguese (Pt), Slovene (Sl), Spanish (Es), Swedish (Sv), and Turkish (Tr). The data come from the CoNLL-X and CoNLL 2007 shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007). For all the experiments conducted, we trained models on the training section of a language's treebank and tested on the test set. Table 1 shows the number of sentences in the treebanks and the size of fine POS tagsets for each language.

Following standard practice, in unsupervised grammar induction experiments we remove punctuation and then eliminate sentences from the data of length greater than 10.

[5] These are based on the Prague Dependency Treebank (Hajič, 1998) and the Tiger treebank (Brants et al., 2002) respectively.

	Pt	Tr	Bg	Jp	El	Sv	Es	Sl	Nl	Da
Training sentences	9,071	4,997	12,823	17,044	2,705	11,042	3,306	1,534	13,349	5,190
Test sentences	288	623	398	709	197	389	206	402	386	322
Size of POS tagset	22	31	54	80	38	41	47	29	12	25

Table 1: The first two rows show the sizes of the training and test datasets for each language. The third row shows the number of fine POS tags in each language including punctuations.

6.2 Universal POS Tags

Our coarse-grained, universal POS tag set consists of the following 12 tags: NOUN, VERB, ADJ (adjective), ADV (adverb), PRON (pronoun), DET (determiner), ADP (preposition or postposition), NUM (numeral), CONJ (conjunction), PRT (particle), PUNC (punctuation mark) and X (a catch-all for other categories such as abbreviations or foreign words). These follow recent work by Das and Petrov (2011) on unsupervised POS tagging in a multilingual setting with parallel data, and have been described in detail by Petrov et al. (2011).

While there might be some controversy about what an appropriate universal tag set should include, these 12 categories (or a subset) cover the most frequent parts of speech and exist in one form or another in all of the languages that we studied. For each language in our data, a mapping from the fine-grained treebank POS tags to these universal POS tags was constructed manually by Petrov et al. (2011).

6.3 Part-of-Speech Tagging

Our first experimental task is POS tagging, and here we describe the specific details of the model, training and inference and the results attained.

6.3.1 Model

The model is a hidden Markov model (HMM), which has been popular for unsupervised tagging tasks (Merialdo, 1994; Elworthy, 1994; Smith and Eisner, 2005; Berg-Kirkpatrick et al., 2010).[6] We use a bigram model and a locally normalized log-linear parameterization, like Berg-Kirkpatrick et al. (2010). These locally normalized log-linear models can look at various aspects of the observation x given a tag y, or the pair of tags in a transition, incorporating overlapping features. In basic monolingual experiments, we used the same set of features as Berg-Kirkpatrick et al. (2010). For the transition log-linear model, Berg-Kirkpatrick et al. (2010) used only a single indicator feature of a tag pair, essentially equating to a traditional multinomial distribution. For the emission log-linear model, several features were used: an indicator feature conjoining the state y and the word x, a feature checking whether x contains a digit conjoined with the state y, another feature indicating whether x contains a hyphen conjoined with y, whether the first letter of x is upper case along with the state y, and finally indicator features corresponding to suffixes up to length 3 present in x conjoined with the state y.

Since only the unlexicalized transition distributions are common across multiple languages, assuming that they all use a set of universal POS tags, akin to Eq. 4, we can have a multilingual version of the transition distributions, by incorporating supervised helper transition probabilities. Thus, we can write:

$$\theta_{y \to y'} = \sum_{\ell=1}^{L} \beta_{\ell,y} \theta^{(\ell)}_{y \to y'} \tag{11}$$

We use the above expression to replace the transition distributions, obtaining a multilingual mixture version of the model. Here, the transition probabilities $\theta^{(\ell)}_{y \to y'}$ for the ℓth helper language are fixed after being estimated using maximum likelihood estimation on the helper language's treebank.

6.3.2 Training and Inference

We trained both the basic feature-based HMM model as well as the multilingual mixture model by optimizing the following objective function:[7]

$$\mathcal{L}(\boldsymbol{\psi}) \quad = \sum_{i=1}^{N} \log \sum_{\boldsymbol{y}} p(\boldsymbol{x}^{(i)}, \boldsymbol{y} \mid \boldsymbol{\psi}) - C\|\boldsymbol{\psi}\|_2^2$$

[6] HMMs can be understood as a special case of PCFGs.

[7] Note that in the objective function, for brevity, we abuse notation by using $\boldsymbol{\psi}$ for both models – monolingual and multilingual; the latter model is also parameterized by $\boldsymbol{\gamma}$.

Method	Pt	Tr	Bg	Jp	El	Sv	Es	Sl	Nl	Da	Avg
Uniform+DG	45.7	**43.6**	**38.0**	60.4	36.7	37.7	31.8	35.9	43.7	36.2	41.0
Mixture+DG	51.5	38.6	35.8	**61.7**	**38.9**	**39.9**	**40.5**	**36.0**	**50.2**	39.9	**43.3**
DG (B-K et al., 2010)	**53.5**	27.9	34.7	52.3	35.3	34.4	40.0	33.4	45.4	**48.8**	40.6

(a)

Method	Pt	Tr	Bg	Jp	El	Sv	Es	Sl	Nl	Da	Avg
Uniform+DG	83.8	**50.4**	81.3	77.9	80.3	**69.0**	82.3	**82.8**	79.3	82.0	76.9
Mixture+DG	**84.7**	50.0	**82.6**	79.9	80.3	67.0	**83.3**	**82.8**	**80.0**	82.0	**77.3**
DG (B-K et al., 2010)	75.4	**50.4**	80.7	**83.4**	**88.0**	61.5	82.3	75.6	79.2	**82.3**	75.9

(b)

Table 2: Results for unsupervised POS induction (a) without a tagging dictionary and (b) with a tag dictionary constructed from the training section of the corresponding treebank. DG (at the bottom) stands for the direct gradient method of Berg-Kirkpatrick et al. (2010) using a monolingual feature-based HMM. "Mixture+DG" is the model where multilingual mixture coefficients $\boldsymbol{\beta}$ of helper languages are estimated using coarse tags (§4), followed by expansion (§5), and then initializing DG with the expanded transition parameters. "Uniform+DG" is the case where $\boldsymbol{\beta}$ are set to 1/4, transitions of helper languages are mixed, expanded, and then DG is initialized with the result. For (a), evaluation is performed using one-to-one mapping accuracy. In case of (b), the tag dictionary solves the problem of tag identification and performance is measured using per word POS accuracy. "Avg" denotes macro-average across the ten languages.

Note that this involves marginalizing out all possible state configurations $\boldsymbol{y}$ for a sentence $\boldsymbol{x}$, resulting in a non-convex objective. As described in §4, we optimized this function using L-BFGS. For the monolingual model, derivatives of the feature weights took the exact same form as Berg-Kirkpatrick et al. (2010), while for the mixture case, we computed gradients with respect to $\boldsymbol{\gamma}$, the unconstrained parameters used to express the mixture coefficients $\boldsymbol{\beta}$ (see Eq. 10). The regularization constant C was set to 1.0 for all experiments, and L-BFGS was run till convergence.

During training, for the basic monolingual feature-based HMM model, we initialized all parameters using small random real values, sampled from $\mathcal{N}(0, 0.01)$. For estimation of the mixture parameters $\boldsymbol{\gamma}$ for our multilingual model (step 3 in §2), we similarly sampled real values from $\mathcal{N}(0, 0.01)$ as an initialization point. Moreover, during this stage, the emission parameters also go through parameter estimation, but they are *monolingual*, and are initialized with real values sampled from $\mathcal{N}(0, 0.01)$; as explained in §2, coarse universal tags are used both in the transitions and emissions during multilingual estimation.

After the mixture parameters $\boldsymbol{\gamma}$ are estimated, we compute the mixture probabilities $\boldsymbol{\beta}$ using Eq. 10. Next, for each tag pair y, y', we compute $\theta_{y \to y'}$, which are the coarse transition probabilities interpolated using $\boldsymbol{\beta}$, given the helper languages. We then expand these transition probabilities (see §5) to result in transition probabilities based on fine tags. Finally, we train a feature-HMM by initializing its transition parameters with natural logarithms of the expanded $\boldsymbol{\theta}$ parameters, and the emission parameters using small random real values sampled from $\mathcal{N}(0, 0.01)$. This implies that the lexicalized emission parameters $\boldsymbol{\eta}$ that were previously estimated in the coarse multilingual model are thrown away and not used for initialization; instead standard initialization is used.

For inference at the testing stage, we use minimum Bayes-risk decoding (or "posterior decoding"), by choosing the most probable tag for each word position, given the entire observation $\boldsymbol{x}$. We chose this strategy because it usually performs slightly better than Viterbi decoding (Cohen and Smith, 2009; Ganchev et al., 2010).

6.3.3 Experimental Setup

For experiments, we considered three configurations, and for each, we implemented two variants of POS induction, one without any kind of supervision, and the other with a tag dictionary. Our baseline is

the direct gradient approach of Berg-Kirkpatrick et al. (2010), which is the current state of the art for this task, outperforming classical HMMs. Because this model achieves strong performance using straightforward MLE, it also serves as the core model within our approach. This model has also been applied in a multilingual setting with parallel data (Das and Petrov, 2011). In this baseline, we set the number of HMM states to the number of fine-grained treebank tags for the given language.

We test two versions of our model. The first initializes training of the target language's POS model using a uniform mixture of the helper language models (i.e., each $\beta_{\ell,y} = \frac{1}{L} = \frac{1}{4}$), and expansion from coarse-grained to fine-grained POS tags as described in §5. We call this model "Uniform+DG."

The second version estimates the mixture coefficients to maximize likelihood, then expands the POS tags (§5), using the result to initialize training of the final model. We call this model "Mixture+DG."

No Tag Dictionary For each of the above configurations, we ran purely unsupervised training without a tag dictionary, and evaluated using *one-to-one mapping* accuracy constraining at most one HMM state to map to a unique treebank tag in the test data, using maximum bipartite matching. This is a variant of the greedy one-to-one mapping scheme of Haghighi and Klein (2006).[8]

With a Tag Dictionary We also ran a second version of each experimental configuration, where we used a tag dictionary to restrict the possible path sequences of the HMM during both learning and inference. This tag dictionary was constructed only from the training section of a given language's treebank. It is widely known that such knowledge improves the quality of the model, though it is an open debate whether such knowledge is realistic to assume. For this experiment we removed punctuation from the training and test data, enabling direct use within the dependency grammar induction experiments.

[8]We also evaluated our approach using the greedy version of this evaluation metric, and results followed the same trends with only minor differences. We did not choose the other variant, *many-to-one mapping* accuracy, because quite often the metric mapped several HMM states to one treebank tag, leaving many treebank tags unaccounted for.

6.3.4 Results

All results for POS induction are shown in Table 2. Without a tag dictionary, in eight out of ten cases, either Uniform+DG or Mixture+DG outperforms the monolingual baseline (Table 2a). For six of these eight languages, the latter model where the mixture coefficients are learned automatically fares better than uniform weighting. *With* a tag dictionary, the multilingual variants outperform the baseline in seven out of ten cases, and the learned mixture outperforms or matches the uniform mixture in five of those seven (Table 2b).

6.4 Dependency Grammar Induction

We next describe experiments for dependency grammar induction. As the basic grammatical model, we adopt the dependency model with valence (Klein and Manning, 2004), which forms the basis for state-of-the-art results for dependency grammar induction in various settings (Cohen and Smith, 2009; Spitkovsky et al., 2010; Gillenwater et al., 2010; Berg-Kirkpatrick and Klein, 2010). As shown in Table 3, DMV obtains much higher accuracy in the supervised setting than the unsupervised setting, suggesting that more can be achieved with this model family.[9] For this reason, and because DMV is easily interpreted as a PCFG, it is our starting point and baseline.

We consider four conditions. The independent variables are (1) whether we use uniform $\boldsymbol{\beta}$ (all set to $\frac{1}{4}$) or estimate them using EM (as described in §4), and (2) whether we simply use the mixture model to decode the test data, or to initialize EM for the DMV. The four settings are denoted "Uniform," "Mixture," "Uniform+EM," and "Mixture+EM."

The results are given in Table 3. In general, the use of data from other languages improves performance considerably; all of our methods outperform the Klein and Manning (2004) initializer, and we achieve state-of-the-art performance for eight out of ten languages. Uniform and Mixture behave similarly, with a slight advantage to the trained mixture setting. Using EM to train the mixture coefficients more often hurts than helps (six languages out of ten). It is well known that likelihood does not cor-

[9]Its supervised performance is still far from the supervised state of the art in dependency parsing.

Method	Pt	Tr	Bg	Jp	El	Sv	Es	Sl	Nl	Da	Avg
Uniform	78.6	45.0	**75.6**	56.3	57.0	**74.0**	73.2	46.1	**50.7**	59.2	61.6
Mixture	76.8	45.3	75.5	58.3	59.5	73.2	75.9	46.0	51.1	**59.9**	**62.2**
Uniform+EM	78.7	43.9	74.7	59.8	**73.0**	70.5	75.5	41.3	45.9	51.3	61.5
Mixture+EM	**79.8**	44.1	72.8	**63.9**	72.3	68.7	**76.7**	41.0	46.0	55.2	62.1
EM (K & M, 2004)	42.5	36.3	54.3	43.0	41.0	42.3	38.1	37.0	38.6	41.4	41.4
PR (G et al., '10)	47.8	**53.4**	54.0	60.2	-	42.2	62.4	**50.3**	37.9	44.0	-
Phylo. (B-K & K, '10)	63.1	-	-	-	-	58.3	63.8	49.6	45.1	41.6	-
Supervised (MLE)	*81.7*	*75.7*	*83.0*	*89.2*	*81.8*	*83.2*	*79.0*	*74.5*	*64.8*	*80.8*	*79.3*

Table 3: Results for dependency grammar induction given gold-standard POS tags, reported as attachment accuracy (fraction of parents which are correct). The three existing methods are: our replication of EM with the initializer from Klein and Manning (2004), denoted "EM"; reported results from Gillenwater et al. (2010) for posterior regularization ("PR"); and reported results from Berg-Kirkpatrick and Klein (2010), denoted "Phylo." "Supervised (MLE)" are oracle results of estimating parameters from gold-standard annotated data using maximum likelihood estimation. "Avg" denotes macro-average across the ten languages.

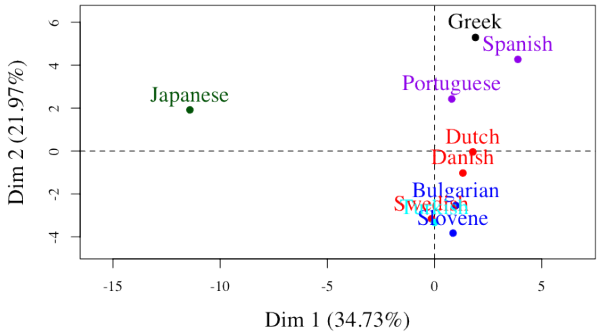


Figure 2: Projection of the learned mixture coefficients through PCA. In green, Japanese. In red, Dutch, Danish and Swedish. In blue, Bulgarian and Slovene. In magenta, Portuguese and Spanish. In black, Greek. In cyan, Turkish.

relate with the true accuracy measurement, and so it is unsurprising that this holds in the constrained mixture family as well. In future work, a different parametrization of the mixture coefficients, through features, or perhaps a Bayesian prior on the weights, might lead to an objective that better simulates accuracy.

Table 3 shows that even uniform mixture coefficients are sufficient to obtain accuracy which supercedes most unsupervised baselines. We were interested in testing whether the coefficients which are learned actually reflect similarities between the languages. To do that, we projected the learned vectors β for each tested language using principal component analysis and plotted the result in Figure 2. It is interesting to note that languages which are closer phylogenetically tend to appear closer to each other in the plot.

Our experiments also show that multilingual learning performs better for dependency grammar induction than part-of-speech tagging. We believe that this happens because of the nature of the models and data we use. The transition matrix in part-of-speech tagging largely depends on word order in the various helper languages, which differs greatly. This means that a mixture of transition matrices will not necessarily yield a meaningful transition matrix. However, for dependency grammar, there are certain universal dependencies which appear in all helper languages, and therefore, a mixture between multinomials for these dependencies still yields a useful multinomial.

6.5 Inducing Dependencies from Words

Finally, we combine the models for POS tagging and grammar induction to perform grammar induction directly from words, instead of gold-standard POS tags. Our approach is as follows:

1. *With* a tag dictionary, learn a fine-grained POS tagging model unsupervised, using either DG or Mixture+DG as described in §6.3 and shown in Table 2b.

Method	Tags	Pt	Tr	Bg	Jp	El	Sv	Es	Sl	Nl	Da	Avg
Joint	DG	**68.4**	**52.4**	62.4	61.4	63.5	**58.2**	67.7	**47.2**	48.3	**50.4**	**57.9**
Joint	Mixture+DG	62.2	47.4	**67.0**	**69.5**	52.2	49.1	**69.3**	36.8	**52.2**	50.1	55.6
Pipeline	DG	60.0	50.8	57.7	64.2	**68.2**	57.9	65.8	45.8	49.9	48.9	56.9
Pipeline	Mixture+DG	59.8	47.1	62.9	68.6	50.0	47.6	68.1	36.4	51.2	48.3	54.0
Gold-standard tags		*79.8*	*45.3*	*75.6*	*63.9*	*73.0*	*74.0*	*76.7*	*46.1*	*50.7*	*59.9*	*64.5*

Table 4: Results for dependency grammar induction over words. "Joint"/"Pipeline" refers to joint/pipeline decoding of tags and dependencies as described in the text. See §6.3 for a description of DG and Mixture+DG. For the induction of dependencies we use the Mixture+EM setting as described in §6.4. All tag induction uses a dictionary as specified in §6.3. The last row in this table indicates the best results using multilingual guidance taken from our methods in Table 3. "Avg" denotes macro-average across the ten languages.

2. Apply the fine-grained tagger to the words in the training data for the dependency parser. We consider two variants: the most probable assignment of tags to words (denoted "Pipeline"), and the posterior distribution over tags for each word, represented as a weighted "sausage" lattice (denoted "Joint"). This idea was explored for joint inference by Cohen and Smith (2007).
3. We apply the Mixture+EM unsupervised parser learning method from §6.4 to the automatically tagged sentences, or the lattices.
4. Given the two models, we infer POS tags on the test data using DG or Mixture+DG to get a lattice (Joint) or a sequence (Pipeline) and then parse using the model from the previous step.[10] The resulting dependency trees are evaluated against the gold standard.

Results are reported in Table 4. In almost all cases, joint decoding of tags and trees performs better than the pipeline. Even though our part-of-speech tagger with multilingual guidance outperforms the completely unsupervised baseline, there is not always an advantage of using this multilingually guided part-of-speech tagger for dependency grammar induction. For Turkish, Japanese, Slovene and Dutch, our unsupervised learner from words outperforms unsupervised parsing using gold-standard part-of-speech tags.

We note that some recent work gives a treatment to unsupervised parsing (but not of dependencies) directly from words (Seginer, 2007). Earlier work that induced part-of-speech tags and then performed unsupervised parsing in a pipeline includes Klein and Manning (2004) and Smith (2006). Headden et al. (2009) described the use of a lexicalized variant of the DMV model, with the use of gold part-of-speech tags.

[10]The decoding method on test data (Joint or Pipeline) was matched to the training method, though they are orthogonal in principle.

7 Conclusion

We presented an approach to exploiting annotated data in helper languages to infer part-of-speech tagging and dependency parsing models in a different, target language, without parallel data. Our approach performs well in many cases. We also described a way to do joint decoding of part-of-speech tags and dependencies which performs better than a pipeline. Future work might consider exploiting a larger number of treebanks, and more powerful techniques for combining models than simple local mixtures.

Acknowledgments

We thank Ryan McDonald and Slav Petrov for helpful comments on an early draft of the paper. This research has been funded by NSF grants IIS-0844507 and IIS-0915187 and by U.S. Army Research Office grant W911NF-10-1-0533.

References

T. Berg-Kirkpatrick and D. Klein. 2010. Phylogenetic grammar induction. In *Proceedings of ACL*.

T. Berg-Kirkpatrick, A. B. Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL-HLT*.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*.

D. Burkett and D. Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*.

S. B. Cohen and N. A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP-CoNLL*.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of HLT-NAACL*.

S. B. Cohen and N. A. Smith. 2010. Covariance in unsupervised learning of probabilistic grammars. *Journal of Machine Learning Research*, 11:3017–3051.

D. Das and S. Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL-HLT*.

D. Elworthy. 1994. Does Baum-Welch re-estimation help taggers? In *Proceedings of ACL*.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of ACL*.

A. Haghighi and D. Klein. 2006. Prototype driven learning for sequence models. In *Proceedings of HLT-NAACL*.

J. Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*. Prague Karolinum, Charles University Press.

W. P. Headden, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of NAACL-HLT*.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of NAACL*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*.

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

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19.

R. McDonald, S. Petrov, and K. Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of EMNLP*.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Compulational Lingustics*, 20(2):155–72.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Mana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In *Building and using Parsed Corpora*, Language and Speech Series. Kluwer, Dordrecht.

T. Naseem, B. Snyder, J. Eisenstein, and R. Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *JAIR*, 36.

T. Naseem, H. Chen, R. Barzilay, and M. Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of EMNLP*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of CoNLL*.

S. Petrov, D. Das, and R. McDonald. 2011. A universal part-of-speech tagset. *ArXiv:1104.2086*.

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of ACL*.

N. A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*.

D. A. Smith and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of EMNLP*.

D. A. Smith and N. A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of EMNLP*.

N. A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University.

B. Snyder and R. Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL*.

B. Snyder, T. Naseem, and R. Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of ACL-IJCNLP*.

V. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Proceedings of NAACL*.

C. Xi and R. Hwa. 2005. A backoff model for bootstrapping resources for non-English languages. In *Proceedings of HLT-EMNLP*.

D. Yarowsky and G. Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of NAACL*.

Multi-Source Transfer of Delexicalized Dependency Parsers

Ryan McDonald
Google
New York, NY
ryanmcd@google.com

Slav Petrov
Google
New York, NY
slav@google.com

Keith Hall
Google
Zürich
kbhall@google.com

Abstract

We present a simple method for transferring dependency parsers from source languages with labeled training data to target languages without labeled training data. We first demonstrate that delexicalized parsers can be directly transferred between languages, producing significantly higher accuracies than unsupervised parsers. We then use a constraint driven learning algorithm where constraints are drawn from parallel corpora to project the final parser. Unlike previous work on projecting syntactic resources, we show that simple methods for introducing multiple source languages can significantly improve the overall quality of the resulting parsers. The projected parsers from our system result in state-of-the-art performance when compared to previously studied unsupervised and projected parsing systems across eight different languages.

1 Introduction

Statistical parsing has been one of the most active areas of research in the computational linguistics community since the construction of the Penn Treebank (Marcus et al., 1993). This includes work on phrase-structure parsing (Collins, 1997; Charniak, 2000; Petrov et al., 2006), dependency parsing (McDonald et al., 2005; Nivre et al., 2006) as well as a number of other formalisms (Clark and Curran, 2004; Wang and Harper, 2004; Shen and Joshi, 2008). As underlying modeling techniques have improved, these parsers have begun to converge to high levels of accuracy for English newswire text. Subsequently, researchers have begun to look at both porting these parsers to new domains (Gildea, 2001; McClosky et al., 2006; Petrov et al., 2010) and constructing parsers for new languages (Collins et al., 1999; Buchholz and Marsi, 2006; Nivre et al., 2007).

One major obstacle in building statistical parsers for new languages is that they often lack the manually annotated resources available for English. This observation has led to a vast amount of research on unsupervised grammar induction (Carroll and Charniak, 1992; Klein and Manning, 2004; Smith and Eisner, 2005; Cohen and Smith, 2009; Berg-Kirkpatrick and Klein, 2010; Naseem et al., 2010; Spitkovsky et al., 2010; Blunsom and Cohn, 2010). Grammar induction systems have seen large advances in quality, but parsing accuracies still significantly lag behind those of supervised systems. Furthermore, they are often trained and evaluated under idealized conditions, e.g., only on short sentences or assuming the existence of gold-standard part-of-speech (POS) tags.[1] The reason for these assumptions is clear. Unsupervised grammar induction is difficult given the complexity of the analysis space. These assumptions help to give the model traction.

The study of unsupervised grammar induction has many merits. Most notably, it increases our understanding of how computers (and possibly humans) learn in the absence of any explicit feedback. However, the gold POS tag assumption weakens any conclusions that can be drawn, as part-of-speech are also a form of syntactic analysis, only shallower. Furthermore, from a practical standpoint, it is rarely the case that we are completely devoid of resources for most languages. This point has been made by

[1] A notable exception is the work of Seginer (2007).

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 62–72,
Edinburgh, Scotland, UK, July 27–31, 2011.

studies that transfer parsers to new languages by projecting syntax across word alignments extracted from parallel corpora (Hwa et al., 2005; Ganchev et al., 2009; Smith and Eisner, 2009). Although again, most of these studies also assume the existence of POS tags.

In this work we present a method for creating dependency parsers for languages for which no labeled training data is available. First, we train a source side English parser that, crucially, is delexicalized so that its predictions rely soley on the part-of-speech tags of the input sentence, in the same vein as Zeman and Resnik (2008). We empirically show that directly transferring delexicalized models (i.e. parsing a foreign language POS sequence with an English parser) already outperforms state-of-the-art unsupervised parsers by a significant margin. This result holds in the presence of both gold POS tags as well as automatic tags projected from English. This emphasizes that even for languages with no syntactic resources – or possibly even parallel data – simple transfer methods can already be more powerful than grammar induction systems.

Next, we use this delexicalized English parser to seed a perceptron learner for the target language. The model is trained to update towards parses that are in high agreement with a source side English parse based on constraints drawn from alignments in the parallel data. We use the augmented-loss learning procedure (Hall et al., 2011) which is closely related to constraint driven learning (Chang et al., 2007; Chang et al., 2010). The resulting parser consistently improves on the directly transferred delexicalized parser, reducing relative errors by 8% on average, and as much as 18% on some languages. Finally, we show that by transferring parsers from multiple source languages we can further reduce errors by 16% over the directly transferred English baseline. This is consistent with previous work on multilingual part-of-speech (Snyder et al., 2009) and grammar (Berg-Kirkpatrick and Klein, 2010; Cohen and Smith, 2009) induction, that shows that adding languages leads to improvements.

We present a comprehensive set of experiments on eight Indo-European languages for which a significant amount of parallel data exists. We make no language specific enhancements in our experiments. We report results for sentences of *all* lengths, as well as with gold and automatically induced part-of-speech tags. We also report results on sentences of length 10 or less with gold part-of-speech tags to compare with previous work. Our results consistently outperform the previous state-of-the-art across all languages and training configurations.

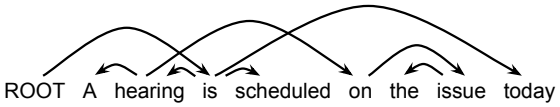


Figure 1: An example (unlabeled) dependency tree.

2 Preliminaries

In this paper we focus on transferring dependency parsers between languages. A dependency parser takes a tokenized input sentence (optionally part-of-speech tagged) and produces a connected tree where directed arcs represent a syntactic head-modifier relationship. An example of such a tree is given in Figure 1. Dependency tree arcs are often labeled with the role of the syntactic relationship, e.g., *is* to *hearing* might be labeled as SUBJECT. However, we focus on unlabeled parsing in order to reduce problems that arise due to different treebank annotation schemes. Of course, even for unlabeled dependencies, significant variations in the annotation schemes remain. For example, in the Danish treebank determiners govern adjectives and nouns in noun phrases, while in most other treebanks the noun is the head of the noun phrase. Unlike previous work (Zeman and Resnik, 2008; Smith and Eisner, 2009), we do not apply any transformations to the treebanks, which makes our results easier to reproduce, but systematically underestimates accuracy.

2.1 Data Sets

The treebank data in our experiments are from the CoNLL shared-tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007). We use English (en) only as a source language throughout the paper. Additionally, we use the following eight languages as both source and target languages: Danish (da), Dutch (nl), German (de), Greek (el), Italian (it), Portuguese (pt), Spanish (es) and Swedish (sv). For languages that were included in both the 2006 and 2007 tasks, we used the treebank from the lat-

ter. We focused on this subset of languages because they are Indo-European and a significant amount of parallel data exists for each language. By presenting results on eight languages our study is already more comprehensive than most previous work in this area. However, the restriction to Indo-European languages does make the results less conclusive when one wishes to transfer a parser from English to Chinese, for example. To account for this, we report additional results in the discussion for non-Indo-European languages. For all data sets we used the predefined training and testing splits.

Our approach relies on a consistent set of part-of-speech tags across languages and treebanks. For this we used the universal tagset from Petrov et al. (2011), which includes: NOUN (nouns), VERB (verbs), ADJ (adjectives), ADV (adverbs), PRON (pronouns), DET (determiners), ADP (prepositions or postpositions), NUM (numerals), CONJ (conjunctions), PRT (particles), PUNC (punctuation marks) and X (a catch-all tag). Similar tagsets are used by other studies on grammar induction and projection (Naseem et al., 2010; Zeman and Resnik, 2008). For all our experiments we replaced the language specific part-of-speech tags in the treebanks with these universal tags.

Like all treebank projection studies we require a corpus of parallel text for each pair of languages we study. For this we used the Europarl corpus version 5 (Koehn, 2005). The corpus was preprocessed in standard ways and word aligned by running six iterations of IBM Model 1 (Brown et al., 1993), followed by six iterations of the HMM model (Vogel et al., 1996) in both directions. We then intersect word alignments to generate one-to-one alignments.

2.2 Parsing Model

All of our parsing models are based on the transition-based dependency parsing paradigm (Nivre, 2008). Specifically, all models use an arc-eager transition strategy and are trained using the averaged perceptron algorithm as in Zhang and Clark (2008) with a beam size of 8. The features used by all models are: the part-of-speech tags of the first four words on the buffer and of the top two words on the stack; the word identities of the first two words on the buffer and of the top word on the stack; the word identity of the syntactic head of the top word on the stack (if available). All feature conjunctions are included. For treebanks with non-projective trees we use the pseudo-projective parsing technique to transform the treebank into projective structures (Nivre and Nilsson, 2005). We focus on using this parsing system for two reasons. First, the parser is near state-of-the-art on English parsing benchmarks and second, and more importantly, the parser is extremely fast to train and run, making it easy to run a large number of experiments. Preliminary experiments using a different dependency parser – MSTParser (McDonald et al., 2005) – resulted in similar empirical observations.

2.3 Evaluation

All systems are evaluated using unlabeled attachment score (UAS), which is the percentage of words (ignoring punctuation tokens) in a corpus that modify the correct head (Buchholz and Marsi, 2006). Furthermore, we evaluate with both gold-standard part-of-speech tags, as well as predicted part-of-speech tags from the projected part-of-speech tagger of Das and Petrov (2011).[2] This tagger relies only on labeled training data for English, and achieves accuracies around 85% on the languages that we consider. We evaluate in the former setting to compare to previous studies that make this assumption. We evaluate in the latter setting to measure performance in a more realistic scenario – when no target language resources are available.

3 Transferring from English

To simplify discussion, we first focus on the most common instantiation of parser transfer in the literature: transferring from English to other languages. In the next section we expand our system to allow for the inclusion of multiple source languages.

3.1 Direct Transfer

We start with the observation that discriminatively trained dependency parsers rely heavily on part-of-speech tagging features. For example, when training and testing a parser on our English data, a parser with all features obtains an UAS of 89.3%[3] whereas

[2] Available at http://code.google.com/p/pos-projection/

[3] The best system at CoNLL 2007 achieved 90.1% and used a richer part-of-speech tagset (Nivre et al., 2007).

a *delexicalized* parser – a parser that only has non-lexical features – obtains an UAS of 82.5%. The key observation is that part-of-speech tags contain a significant amount of information for unlabeled dependency parsing.

This observation combined with our universal part-of-speech tagset, leads to the idea of *direct transfer*, i.e., directly parsing the target language with the source language parser without relying on parallel corpora. This idea has been previously explored by Zeman and Resnik (2008) and recently by Søgaard (2011). Because we use a mapping of the treebank specific part-of-speech tags to a common tagset, the performance of a such a system is easy to measure – simply parse the target language data set with a delexicalized parser trained on the source language data. We conducted two experiments. In the first, we assumed that the test set for each target language had gold part-of-speech tags, and in the second we used predicted part-of-speech tags from the projection tagger of Das and Petrov (2011), which also uses English as the source language.

UAS for all sentence lengths without punctuation are given in Table 1. We report results for both the English direct transfer parser (en-dir.) as well as a baseline unsupervised grammar induction system – the dependency model with valence (DMV) of Klein and Manning (2004), as obtained by the implementation of Ganchev et al. (2010). We trained on sentences of length 10 or less and evaluated on all sentences from the test set.[4] For DMV, we reversed the direction of all dependencies if this led to higher performance. From this table we can see that direct transfer is a very strong baseline and is over 20% absolute better than the DMV model for both gold and predicted POS tags. Table 4, which we will discuss in more detail later, further shows that the direct transfer parser also significantly outperforms state-of-the-art unsupervised grammar induction models, but in a more limited setting of sentences of length less than 10.

Direct transfer works for a couple of reasons. First, part-of-speech tags contain a significant amount of information for parsing unlabeled dependencies. Second, this information can be transferred, to some degree, across languages and treebank standards. This is because, at least for Indo-European languages, there is some regularity in how syntax is expressed, e.g., primarily SVO, prepositional, etc. Even though there are some differences with respect to relative location of certain word classes, strong head-modifier POS tag preferences can still help resolve these, especially when no other viable alternatives are available. Consider for example an artificial sentence with a tag sequence: 'VERB NOUN ADJ DET PUNC'. The English parser still predicts that the NOUN and PUNC modify the VERB and the ADJ and DET modify the NOUN, even though in the English data such noun phrases are unlikely.[5]

3.2 Projected Transfer

Unlike most language transfer systems for parsers, the direct transfer approach does not rely on projecting syntax across aligned parallel corpora (modulo the fact that non-gold tags come from a system that uses parallel corpora). In this section we describe a simple mechanism for projecting from the direct transfer system using large amounts of parallel data in a similar vein to Hwa et al. (2005), Ganchev et al. (2009), Smith and Eisner (2009) inter alia. The algorithm is based on the work of Hall et al. (2011) for training extrinsic parser objective functions and borrows heavily from ideas in learning with weak supervision including work on learning with constraints (Chang et al., 2007) and posterior regularization (Ganchev et al., 2010). In our case, the weak signals come from aligned source and target sentences, and the agreement in their corresponding parses, which is similar to posterior regularization or the bilingual view of Smith and Smith (2004) and Burkett et al. (2010).

The algorithm is given in Figure 2. It starts by labeling a set of target language sentences with a parser, which in our case is the direct transfer parser from the previous section (line 1). Next, it uses these parsed target sentences to 'seed' a new parser by training a parameter vector using the predicted parses as a gold standard via standard perceptron updates for J rounds (lines 3-6). This generates a parser that emulates the direct transfer parser, but

[4] Training on all sentences results in slightly lower accuracies on average.

[5] This requires a transition-based parser with a beam greater than 1 to allow for ambiguity to be resolved at later stages.

Notation:

x:	input sentence
y:	dependency tree
a:	alignment
w:	parameter vector
$\phi(x, y)$:	feature vector
DP:	dependency parser, i.e., $DP : x \rightarrow y$

Input:

$\mathcal{X} = \{x_i\}_{i=1}^{n}$:	target language sentences
$\mathcal{P} = \{(x_i^s, x_i^t, a_i)\}_{i=1}^{m}$:	aligned source-target sentences
DP_{delex}:	delexicalized source parser
DP_{lex}:	lexicalized source parser

Algorithm:

1. Let $\mathcal{X}' = \{(x_i, y_i)\}_{i=1}^{n}$ where $y_i = DP_{\text{delex}}(x_i)$
2. $w = 0$

seed-stage

3. for $j : 1 \ldots J$
4. for $x_i : x_1 \ldots x_n$
5. Let $y = \text{argmax}_y \, w \cdot \phi(x_i, y)$
6. $w = w + \phi(x_t, y_i) - \phi(x_i, y)$

projection-stage

7. for $(x_i^s, x_i^t, a_i) : (x_1^s, x_1^t, a_1) \ldots (x_m^s, x_m^s, a_m)$
8. Let $y_s = DP_{\text{lex}}(x_i^s)$
9. Let $\mathcal{Y}_t = \{y_i^1, \ldots, y_i^k\}$, where:
   $y_i^k = \text{argmax}_{y \notin \{y_i^1, \ldots, y_i^{k-1}\}} \, w \cdot \phi(x_i^t, y)$
10. Let $y_t = \text{argmax}_{y_t \in \mathcal{Y}_t} \text{ALIGN}(y_s, y_t, a_i)$
11. $w = w + \phi(x_i, y_t) - \phi(x_i, y_i^1)$

return DP^* such that $DP^*(x) = \text{argmax}_y \, w \cdot \phi(x, y)$

Figure 2: Perceptron-based learning algorithm for training a parser by seeding the model with a direct transfer parser and projecting constraints across parallel corpora.

has now been lexicalized and is working in the space of target language sentences. Next, the algorithm iterates over the sentences in the parallel corpus. It parses the English sentence with an English parser (line 8, again a lexicalized parser). It then uses the current target language parameter vector to create a k-best parse list for the target sentence (line 9). From this list, it selects the parse whose dependencies align most closely with the English parse via the pre-specified alignment (line 10, also see below for the definition of the ALIGN function). It then uses this selected parse as a proxy to the gold standard parse to update the parameters (line 11).

The intuition is simple. The parser starts with non-random accuracies by emulating the direct transfer model and slowly tries to induce better parameters by selecting parses from its k-best list that are considered 'good' by some external metric. The algorithm then updates towards that output. In this case 'goodness' is determined through the pre-specified sentence alignment and how well the target language parse aligns with the English parse. As a result, the model will, ideally, converge to a state where it predicts target parses that align as closely as possible with the corresponding English parses. However, since we seed the learner with the direct transfer parser, we bias the parameters to select parses that both align well and also have high scores under the direct transfer model. This helps to not only constrain the search space at the start of learning, but also helps to bias dependencies between words that are not part of the alignment.

So far we have not defined the ALIGN function that is used to score potential parses. Let $a = \{(s_{(1)}, t_{(1)}), \ldots, (s_{(n)}, t_{(n)})\}$ be an alignment where $s_{(i)}$ is a word in the source sentence x_s (not necessarily the i^{th} word) and $t_{(i)}$ is similarly a word in the target sentence x_t (again, not necessarily the i^{th} word). The notation $(s_{(i)}, t_{(i)}) \in a$ indicates two words are the i^{th} aligned pair in a. We define the ALIGN function to encode the Direct Correspondence Assumption (DCA) from Hwa et al. (2005):

$$\text{ALIGN}(y_s, y_t, a) = \sum_{\substack{(s_{(i)}, t_{(i)}) \in a \\ (s_{(j)}, t_{(j)}) \in a}} \text{SCORE}(y_s, y_t, (s_{(i)}, s_{(j)}), (t_{(i)}, t_{(j)}))$$

$$\text{SCORE}(y_s, y_t, (s_{(i)}, s_{(j)}), (t_{(i)}, t_{(j)})) = \begin{cases} +1 & \text{if } (s_{(i)}, s_{(j)}) \in y_s \text{ and } (t_{(i)}, t_{(j)}) \in y_t \\ -1 & \text{if } (s_{(i)}, s_{(j)}) \in y_s \text{ and } (t_{(i)}, t_{(j)}) \notin y_t \\ -1 & \text{if } (s_{(i)}, s_{(j)}) \notin y_s \text{ and } (t_{(i)}, t_{(j)}) \in y_t \\ 0 & \text{otherwise} \end{cases}$$

The notation $(i, j) \in y$ indicates that a dependency from head i to modifier j is in tree y. The ALIGN function rewards aligned head-modifier pairs and penalizes unaligned pairs when a possible alignment exists. For all other cases it is agnostic, i.e., when one or both of the modifier or head are not aligned.

Figure 3 shows an example of aligned English-Greek sentences, the English parse and a potential Greek parse. In this case the ALIGN function returns a value of 2. This is because there are three aligned dependencies: *took*→*book*, *book*→*the* and

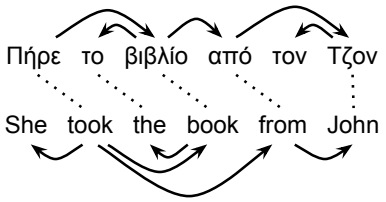


Figure 3: A Greek and English sentence pair. Word alignments are shown as dashed lines, dependency arcs as solid lines.

from→John. These add 3 to the score. There is one incorrectly aligned dependency: the preposition mistakenly modifies the noun on the Greek side. This subtracts 1. Finally, there are two dependencies that do not align: the subject on the English side and a determiner to a proper noun on the Greek side. These do not effect the result.

The learning algorithm in Figure 2 is an instance of augmented-loss training (Hall et al., 2011) which is closely related to the constraint driven learning algorithms of Chang et al. (2007). In that work, external constraints on output structures are used to help guide the learner to good parameter regions. In our model, we use constraints drawn from parallel data exactly in the same manner. Since posterior regularization is closely related to constraint driven learning, this makes our algorithm also similar to the parser projection approach of Ganchev et al. (2009). There are a couple of differences. First, we bias our model towards the direct transfer model, which is already quite powerful. Second, our alignment constraints are used to select parses from a k-best list, whereas in posterior regularization they are used as soft constraints on full model expectations during training. The latter is beneficial as the use of k-best lists does not limit the class of parsers to those whose parameters and search space decompose neatly with the DCA loss function. An empirical comparison to Ganchev et al. (2009) is given in Section 5.

Results are given in Table 1 under the column *en-proj*. For all experiments we train the seed-stage perceptron for 5 iterations ($J = 5$) and we use one hundred times as much parallel data as seed stage non-parallel data ($m = 100n$). The seed-stage non-parallel data is the training portion of each treebank, stripped of all dependency annotations. After training the projected parser we average the parameters

	gold-POS			pred-POS		
	DMV	en-dir.	en-proj.	DMV	en-dir.	en-proj.
da	33.4	45.9	48.2	18.4	44.0	45.5
de	18.0	47.2	50.9	30.3	44.7	47.4
el	39.9	63.9	66.8	21.2	63.0	65.2
es	28.5	53.3	55.8	19.9	50.2	52.4
it	43.1	57.7	60.8	37.7	53.7	56.3
nl	38.5	60.8	67.8	19.9	62.1	66.5
pt	20.1	69.2	71.3	21.0	66.2	67.7
sv	44.0	58.3	61.3	33.8	56.5	59.7
avg	33.2	57.0	60.4	25.3	55.0	57.6

Table 1: UAS for the unsupervised DMV model (DMV), a delexicalized English direct transfer parser (en-dir.) and a English projected parser (en-proj.). Measured on all sentence lengths for both gold and predicted part-of-speech tags as input.

of the model (Collins, 2002). The parsers evaluated using predicted part-of-speech tags use the predicted tags at both training and testing time and are thus free of any target language specific resources.

When compared with the direct transfer model (en-dir. in Table 1), we can see that there is an improvement for every single language, reducing relative error by 8% on average (57.0% to 60.4%) and up to 18% for Dutch (60.8 to 67.8%). One could wonder whether the true power of the projection model comes from the re-lexicalization step – lines 3-6 of the algorithm. However, if just this step is run, then the average UAS only increases from 57.0% to 57.4%, showing that most of the improvement comes from the projection stage. Note that the results in Table 1 indicate that parsers using predicted part-of-speech tags are only slightly worse than the parsers using gold tags (about 2-3% absolute), showing that these methods are robust to tagging errors.

4 Multi-Source Transfer

The previous section focused on transferring an English parser to a new target language. However, there are over 20 treebanks available for a variety of language groups including Indo-European, Altaic (including Japanese), Semitic, and Sino-Tibetan. Many of these are even in standardized formats (Buchholz and Marsi, 2006; Nivre et al., 2007). Past studies have shown that for both part-of-speech tagging and grammar induction, learning with multiple comparable languages leads to improvements (Cohen and Smith, 2009; Snyder et al., 2009; Berg-Kirkpatrick and Klein, 2010). In this section we ex-

		Source Training Language								
		da	de	el	en	es	it	nl	pt	sv
Target Test Language	da	**79.2**	45.2	44.0	45.9	45.0	48.6	46.1	48.1	47.8
	de	34.3	**83.9**	53.2	47.2	45.8	53.4	55.8	55.5	46.2
	el	33.3	52.5	**77.5**	63.9	41.6	59.3	57.3	58.6	47.5
	en	34.4	37.9	45.7	**82.5**	28.5	38.6	43.7	42.3	43.7
	es	38.1	49.4	57.3	53.3	**79.7**	68.4	51.2	66.7	41.4
	it	44.8	56.7	66.8	57.7	64.7	**79.3**	57.6	69.1	50.9
	nl	38.7	43.7	62.1	60.8	40.9	50.4	**73.6**	58.5	44.2
	pt	42.5	52.0	66.6	69.2	68.5	74.7	67.1	**84.6**	52.1
	sv	44.5	57.0	57.8	58.3	46.3	53.4	54.5	66.8	**84.8**

Table 2: UAS for all source-target language pairs. Each column represents which source language was used to train a delexicalized parser and each row represents which target language test data was used. Bold numbers are when source equals target and underlined numbers are the single best UAS for a target language. Results are for all sentence lengths without punctuation.

amine whether this is also true for parser transfer.

Table 2 shows the matrix of source-target language UAS for all nine languages we consider (the original eight target languages plus English). We can see that there is a wide range from 33.3% to 74.7%. There is also a wide range of values depending on the source training data and/or target testing data, e.g., Portuguese as a source tends to parse target languages much better than Danish, and is also more amenable as a target testing language. Some of these variations are expected, e.g., the Romance languages (Spanish, Italian and Portuguese) tend to transfer well to one another. However, some are unexpected, e.g., Greek being the best source language for Dutch, as well as German being one of the worst. This is almost certainly due to different annotation schemes across treebanks. Overall, Table 2 does indicate that there are possible gains in accuracy through the inclusion of additional languages.

In order to take advantage of treebanks in multiple languages, our multi-source system simply concatenates the training data from all non-target languages. In other words, the multi-source direct transfer parser for Danish will be trained by first concatenating the training corpora of the remaining eight languages, training a delexicalized parser on this data and then directly using this parser to analyze the Danish test data. For the multi-source projected parser, the procedure is identical to that in Section 3.2 except that we use the multi-source direct transfer model to seed the algorithm instead of the English-only direct transfer model. For these experiments we still only use English-target parallel data because that is the format of the readily available data in the Europarl corpus.

Table 3 presents four sets of results. The first (best-source) is the direct transfer results for the oracle single-best source language per target language. The second (avg-source) is the mean UAS over all source languages per target language. The third (multi-dir.) is the multi-source direct transfer system. The fourth and final result set (multi-proj.) is the multi-source projected system. The resulting parsers are typically much more accurate than the English direct transfer system (Table 1). On average, the multi-source direct transfer system reduces errors by 10% relative over the English-only direct transfer system. These improvements are not consistent. For Greek and Dutch we see significant losses relative to the English-only system. An inspection of Table 2 shows that for these two languages English is a particularly good source training language.

For the multi-source projected system the results are mixed. Some languages see basically no change relative the multi-source direct transfer model, while some languages see modest to significant increases. But again, there is an overall trend to better models. In particular, starting with an English-only direct transfer parser with 57.0% UAS on average, by adding parallel corpora and multiple source languages we finish with parser having 63.8% UAS on average, which is a relative reduction in error of roughly 16% and more than doubles the performance of a DMV model (Table 1).

Interestingly, the multi-source systems provide, on average, accuracies near that of the single-best source language and significantly better than the average source UAS. Thus, even this simple method of

	best-source		avg-source	gold-POS		pred-POS	
	source	gold-POS	gold-POS	multi-dir.	multi-proj.	multi-dir.	multi-proj.
da	it	48.6	46.3	48.9	49.5	46.2	47.5
de	nl	55.8	48.9	56.7	56.6	51.7	52.0
el	en	63.9	51.7	60.1	65.1	58.5	63.0
es	it	68.4	53.2	64.2	64.5	55.6	56.5
it	pt	69.1	58.5	64.1	65.0	56.8	58.9
nl	el	62.1	49.9	55.8	65.7	54.3	64.4
pt	it	74.8	61.6	74.0	75.6	67.7	70.3
sv	pt	66.8	54.8	65.3	68.0	58.3	62.1
avg		63.7	51.6	61.1	63.8	56.1	59.3

Table 3: UAS for multi-source direct (multi-dir.) and projected (multi-proj.) transfer systems. *best-source* is the best source model from the languages in Table 2 (excluding the target language). *avg-source* is the mean UAS over the source models for the target (excluding target language).

multi-source transfer already provides strong performance gains. We expect that more principled techniques will lead to further improvements. For example, recent work by Søgaard (2011) explores data set sub-sampling methods. Unlike our work, Søgaard found that simply concatenating all the data led to degradation in performance. Cohen et al. (2011) explores the idea learning language specific mixture coefficients for models trained independently on the target language treebanks. However, their results show that this method often did not significantly outperform uniform mixing.

5 Comparison

Comparing unsupervised and parser projection systems is difficult as many publications use non-overlapping sets of languages or different evaluation criteria. We compare to the following three systems that do not augment the treebanks and report results for some of the languages that we considered:

- **USR**: The weakly supervised system of Naseem et al. (2010), in which manually defined universal syntactic rules (USR) are used to constrain a probabilistic Bayesian model. In addition to their original results, we also report results using the same part-of-speech tagset as the systems described in this paper (**USR**†). This is useful for two reasons. First, it makes the comparison more direct. Second, we can generate USR results for all eight languages and not just for the languages that they report.

- **PGI**: The phylogenetic grammar induction (PGI) model of Berg-Kirkpatrick and Klein (2010), in which the parameters of completely unsupervised DMV models for multiple languages are coupled via a phylogenetic prior.

- **PR**: The posterior regularization (PR) approach of Ganchev et al. (2009), in which a supervised English parser is used to generate constraints that are projected using a parallel corpus and used to regularize a target language parser. We report results without treebank specific rules.

Table 4 gives results comparing the models presented in this work to those three systems. For this comparison we use sentences of length 10 or less after punctuation has been removed in order to be consistent with reported results. The overall trends carry over from the full treebank setting to this reduced sentence length setup: the projected models outperform the direct transfer models and multi-source transfer gives higher accuracy than transferring only from English. Most previous work has assumed gold part-of-speech tags, but as the code for USR is publicly available we were able to train it using the same projected part-of-speech tags used in our models. These results are also given in Table 4 under USR†. Again, we can see that the multi-source systems (both direct and projected) significantly outperform the unsupervised models.

It is not surprising that a parser transferred from annotated resources does significantly better than unsupervised systems since it has much more information from which to learn. The PR system of Ganchev et al. (2009) is similar to ours as it also projects syntax across parallel corpora. For Spanish we can see that the multi-source direct transfer parser is better (75.1% versus 70.6%), and this is also true for the multi-source projected parser

	⟵ gold-POS ⟶								← pred-POS →		
	en-dir.	en-proj.	multi-dir.	multi-proj.	USR†	USR	PGI	PR	multi-dir.	multi-proj.	USR†
da	53.2	57.4	58.4	58.8	55.1	51.9	41.6		54.9	54.6	41.7
de	65.9	67.0	74.9	72.0	60.0				63.7	63.4	55.1
el	73.9	73.9	73.5	78.7	60.3				65.2	74.3	53.4
es	58.0	62.3	75.1	73.2	68.3	67.2	58.4	70.6	59.1	56.8	43.3
it	65.5	69.9	75.5	75.5	47.9				65.5	70.2	41.4
nl	67.6	72.2	58.8	70.7	44.0		45.1		56.3	67.2	38.8
pt	77.9	80.6	81.1	86.2	70.9	71.5	63.0		74.0	79.2	66.4
sv	70.4	71.3	76.0	77.6	52.6		58.3		72.0	73.9	59.4
avg	66.6	69.4	71.7	74.1	57.4				63.9	67.5	49.9

Table 4: UAS on sentences of length 10 or less without punctuation, comparing the systems presented in this work to three representative systems from related work. en-dir./en-proj. are the direct/projected English parsers and multi-dir./multi-proj. are the multi-source direct/projected parsers. Section 5 contains a description of the baseline systems.

(73.2%). Ganchev et al. also report results for Bulgarian. We trained a multi-source direct transfer parser for Bulgarian which obtained a score of 72.8% versus 67.8% for the PR system. If we only use English as a source language, as in Ganchev et al., the English direct transfer model achieves 66.1% on Bulgarian and 69.3% on Spanish versus 67.8% and 70.6% for PR. In this setting the English projected model gets 72.0% on Spanish. Thus, under identical conditions the direct transfer model obtains accuracies comparable to PR.[6]

Another projection based system is that of Smith and Eisner (2009), who report results for German (68.5%) and Spanish (64.8%) on sentences of length 15 and less inclusive of punctuation. Smith and Eisner use custom splits of the data and modify a subset of the dependencies. The multi-source projected parser obtains 71.9% for German and 67.8% for Spanish on this setup.[7] If we cherry-pick the source language the results can improve, e.g., for Spanish we can obtain 71.7% and 70.8% by directly transferring parsers form Italian or Portuguese respectively.

6 Discussion

One fundamental point the above experiments illustrate is that even for languages for which no resources exist, simple methods for transferring parsers work remarkably well. In particular, if one can transfer part-of-speech tags, then a large part of transferring unlabeled dependencies has been solved. This observation should lead to a new baseline in unsupervised and projected grammar induction – the UAS of a delexicalized English parser. Of course, our experiments focus strictly on Indo-European languages. Preliminary experiments for Arabic (ar), Chinese (zh), and Japanese (ja) suggest similar direct transfer methods are applicable. For example, on the CoNLL test sets, a DMV model obtains UAS of 28.7/41.8/34.6% for ar/zh/ja respectively, whereas an English direct transfer parser obtains 32.1/53.8/32.2% and a multi-source direct transfer parser obtains 39.9/41.7/43.3%. In this setting only Indo-European languages are used as source data. Thus, even across language groups direct transfer is a reasonable baseline. However, this is not necessary as treebanks are available for a number of language groups, e.g., Indo-European, Altaic, Semitic, and Sino-Tibetan.

The second fundamental observation is that when available, multiple sources should be used. Even through naive multi-source methods (concatenating data), it is possible to build a system that has comparable accuracy to the single-best source for all languages. This advantage does not come simply from having more data. In fact, if we randomly sampled from the multi-source data until the training set size was equivalent to the size of the English data, then the results still hold (and in fact go up slightly for some languages). This suggests that even better transfer models can be produced by separately weighting each of the sources depending on the target language – either weighting by hand, if we know the language group of the target language, or auto-

[6] Note that the last set of results was obtained by using the same English training data as Ganchev et al. Using the CoNLL 2007 English data set for training, the English direct transfer model is 63.2% for Bulgarian and 58.0% for Spanish versus 67.8% and 70.6% for PR, highlighting the large impact that difference treebank annotation standards can have.

[7] Data sets and evaluation criteria obtained via communications with David Smith and Jason Eisner.

matically, if we do not. As previously mentioned, the latter has been explored in both Søgaard (2011) and Cohen et al. (2011).

7 Conclusions

We presented a simple, yet effective approach for projecting parsers from languages with labeled training data to languages without any labeled training data. Central to our approach is the idea of delexicalizing the models, which combined with a standardized part-of-speech tagset allows us to directly transfer models between languages. We then use a constraint driven learning algorithm to adapt the transferred parsers to the respective target language, obtaining an additional 16% error reduction on average in a multi-source setting. Our final parsers achieve state-of-the-art accuracies on eight Indo-European languages, significantly outperforming previous unsupervised and projected systems.

Acknowledgements: We would like to thank Kuzman Ganchev, Valentin Spitkovsky and Dipanjan Das for numerous discussions on this topic and comments on earlier drafts of this paper. We would also like to thank Shay Cohen, Dipanjan Das, Noah Smith and Anders Søgaard for sharing early drafts of their recent related work.

References

T. Berg-Kirkpatrick and D. Klein. 2010. Phylogenetic grammar induction. In *Proc. of ACL*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. *Proc. of EMNLP*.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

D. Burkett, S. Petrov, J. Blitzer, and D. Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *Proc. of CoNLL*.

G. Carroll and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Proc. of the Working Notes of the Workshop Statistically-Based NLP Techniques*.

M.W. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of ACL*.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. 2010. Structured output learning with indirect supervision. In *Proc. of ICML*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of NAACL*.

S. Clark and J. R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proc. of ACL*.

S.B. Cohen and N.A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proc. of NAACL*.

S.B. Cohen, D. Das, and N.A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proc. of EMNLP*.

M. Collins, J. Hajič, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proc. of ACL*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of ACL*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of ACL*.

D. Das and S. Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proc. of ACL-HLT*.

K. Ganchev, J. Gillenwater, and B. Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proc. of ACL-IJCNLP*.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

D. Gildea. 2001. Corpus variation and parser performance. In *Proc of EMNLP*.

K. Hall, R. McDonald, J. Katz-Brown, and M. Ringgaard. 2011. Training dependency parsers by jointly optimizing multiple objectives. In *Proc. of EMNLP*.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(03):311–325.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proc. of ACL*.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

M. P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proc. of ACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

T. Naseem, H. Chen, R. Barzilay, and M. Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proc. of EMNLP*.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL*.

J. Nivre, J. Hall, and J. Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proc. of LREC*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of EMNLP-CoNLL*.

J. Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of ACL*.

S. Petrov, P. Chang, M. Ringgaard, and H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *EMNLP '10*.

S. Petrov, D. Das, and R. McDonald. 2011. A universal part-of-speech tagset. In *ArXiv:1104.2086*.

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *Proc. of ACL*.

L. Shen and A.K. Joshi. 2008. Ltag dependency parsing with bidirectional incremental construction. In *Proc. of EMNLP*.

N.A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*.

D.A. Smith and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proc. of EMNLP*.

D.A. Smith and N.A. Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *Proc. of EMNLP*.

B. Snyder, T. Naseem, J. Eisenstein, and R. Barzilay. 2009. Adding more languages improves unsupervised multilingual part-of-speech tagging: A Bayesian nonparametric approach. In *Proc. of NAACL*.

A. Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proc. ACL*.

V.I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Proc. of NAACL-HLT*.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. of COLING*.

W. Wang and M. P. Harper. 2004. A statistical constraint dependency grammar (CDG) parser. In *Proc. of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

D. Zeman and P. Resnik. 2008. Cross-language parser adaptation between related languages. In *NLP for Less Privileged Languages*.

Y. Zhang and S. Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In *Proc. of EMNLP*.

SMT Helps Bitext Dependency Parsing

Wenliang Chen[†‡], **Jun'ichi Kazama**[‡], **Min Zhang**[†], **Yoshimasa Tsuruoka**[*‡], **Yujie Zhang**[⋆‡], **Yiou Wang**[‡], **Kentaro Torisawa**[‡] **and Haizhou Li**[†]
[†]Human Language Technology, Institute for Infocomm Research, Singapore
[‡]National Institute of Information and Communications Technology (NICT), Japan
[*]School of Information Science, JAIST, Japan
[⋆]Beijing Jiaotong University, China
{wechen, mzhang, hli}@i2r.a-star.edu.sg
{kazama, torisawa, yujie, wangyiou}@nict.go.jp
tsuruoka@jaist.ac.jp

Abstract

We propose a method to improve the accuracy of parsing bilingual texts (bitexts) with the help of statistical machine translation (SMT) systems. Previous bitext parsing methods use human-annotated bilingual treebanks that are hard to obtain. Instead, our approach uses an auto-generated bilingual treebank to produce bilingual constraints. However, because the auto-generated bilingual treebank contains errors, the bilingual constraints are noisy. To overcome this problem, we use large-scale unannotated data to verify the constraints and design a set of effective bilingual features for parsing models based on the verified results. The experimental results show that our new parsers significantly outperform state-of-the-art baselines. Moreover, our approach is still able to provide improvement when we use a larger monolingual treebank that results in a much stronger baseline. Especially notable is that our approach can be used in a purely monolingual setting with the help of SMT.

1 Introduction

Recently there have been several studies aiming to improve the performance of parsing bilingual texts (bitexts) (Smith and Smith, 2004; Burkett and Klein, 2008; Huang et al., 2009; Zhao et al., 2009; Chen et al., 2010). In bitext parsing, we can use the information based on "bilingual constraints" (Burkett and Klein, 2008), which do not exist in monolingual sentences. More accurate bitext parsing results can be effectively used in the training of syntax-based machine translation systems (Liu and Huang, 2010).

Most previous studies rely on bilingual treebanks to provide bilingual constraints for bitext parsing. Burkett and Klein (2008) proposed joint models on bitexts to improve the performance on either or both sides. Their method uses bilingual treebanks that have human-annotated tree structures on both sides. Huang et al. (2009) presented a method to train a source-language parser by using the reordering information on words between the sentences on two sides. It uses another type of bilingual treebanks that have tree structures on the source sentences and their human-translated sentences. Chen et al. (2010) also used bilingual treebanks and made use of tree structures on the target side. However, the bilingual treebanks are hard to obtain, partly because of the high cost of human translation. Thus, in their experiments, they applied their methods to a small data set, the manually translated portion of the Chinese Treebank (CTB) which contains only about 3,000 sentences. On the other hand, many large-scale monolingual treebanks exist, such as the Penn English Treebank (PTB) (Marcus et al., 1993) (about 40,000 sentences in Version 3) and the latest version of CTB (over 50,000 sentences in Version 7).

In this paper, we propose a bitext parsing approach in which we produce the bilingual constraints on existing monolingual treebanks with the help of SMT systems. In other words, we aim to improve source-language parsing with the help of automatic translations.

In our approach, we first use an SMT system to translate the sentences of a source monolingual treebank into the target language. Then, the target sentences are parsed by a parser trained on a target monolingual treebank. We then obtain a bilingual treebank that has human annotated trees on the source side and auto-generated trees on the target side. Although the sentences and parse trees on the

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 73–83,
Edinburgh, Scotland, UK, July 27–31, 2011.

target side are not perfect, we expect that we can improve bitext parsing performance by using this newly auto-generated bilingual treebank. We build word alignment links automatically using a word alignment tool. Then we can produce a set of bilingual constraints between the two sides.

Because the translation, parsing, and word alignment are done automatically, the constraints are not reliable. To overcome this problem, we verify the constraints by using large-scale unannotated monolingual sentences and bilingual sentence pairs. Finally, we design a set of bilingual features based on the verified results for parsing models.

Our approach uses existing resources including monolingual treebanks to train monolingual parsers on both sides, bilingual unannotated data to train SMT systems and to extract bilingual subtrees, and target monolingual unannotated data to extract monolingual subtrees. In summary, we make the following contributions:

- We propose an approach that uses an auto-generated bilingual treebank rather than human-annotated bilingual treebanks used in previous studies (Burkett and Klein, 2008; Huang et al., 2009; Chen et al., 2010). The auto-generated bilingual treebank is built with the help of SMT systems.
- We verify the unreliable constraints by using the existing large-scale unannotated data and design a set of effective bilingual features over the verified results. Compared to Chen et al. (2010) that also used tree structures on the target side, our approach defines the features on the auto-translated sentences and auto-parsed trees, while theirs generates the features by some rules on the human-translated sentences.
- Our parser significantly outperforms state-of-the-art baseline systems on the standard test data of CTB containing about 3,000 sentences. Moreover, our approach continues to achieve improvement when we build our system using the latest version of CTB (over 50,000 sentences) that results in a much stronger baseline.
- We show the possibility that we can improve the performance even if the test set has no human translation. This means that our proposed approach can be used in a purely monolingual setting with the help of SMT. To our knowledge, this paper is the first one that demonstrates this widened applicability, unlike the previous studies that assumed that the parser is applied only on the bitexts made by humans.

Throughout this paper, we use Chinese as the source language and English as the target language. The rest of this paper is organized as follows. Section 2 introduces the motivation of this work. Section 3 briefly introduces the parsing model used in the experiments. Section 4 describes a set of bilingual features based on the bilingual constraints and Section 5 describes how to use large-scale unannotated data to verify the bilingual constraints and define another set of bilingual features based on the verified results. Section 6 explains the experimental results. Finally, in Section 7 we draw conclusions.

2 Motivation

Here, bitext parsing is the task of parsing source sentences with the help of their corresponding translations. Figure 1-(a) shows an example of the input of bitext parsing, where ROOT is an artificial root token inserted at the beginning and does not depend on any other token in the sentence, the dashed undirected links are word alignment links, and the directed links between words indicate that they have a dependency relation. Given such inputs, we build dependency trees for the source sentences. Figure 1-(b) shows the output of bitext parsing for the example in 1-(a).

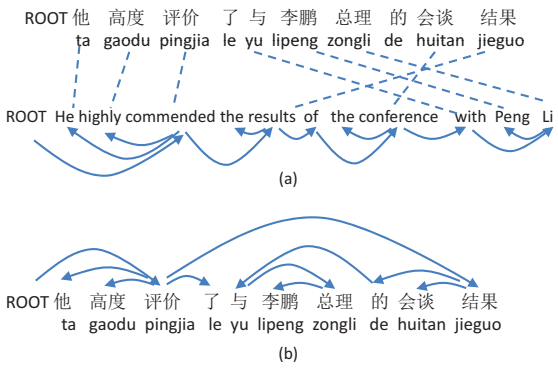


Figure 1: Input and output of our approach

In bitext parsing, some ambiguities exist on the source side, but they may be unambiguous on the

target side. These differences are expected to help improve source-side parsing.

Suppose we have a Chinese sentence shown in Figure 2-(a). In this sentence, there is a nominalization case (Li and Thompson, 1997) in which the particle "的(de)/nominalizer" is placed after the verb compound "培育(peiyu)起来(qilai)/cultivate" to modify "技巧(jiqiao)/skill". This nominalization is a relative clause, but does not have a clue about its boundary. That is, it is very hard to determine which word is the head of "技巧(jiqiao)/skill". The head may be "发挥(fahui)/demonstrate" or "培育(peiyu)/cultivate", as shown in Figure 2-(b) and -(c), where (b) is correct.

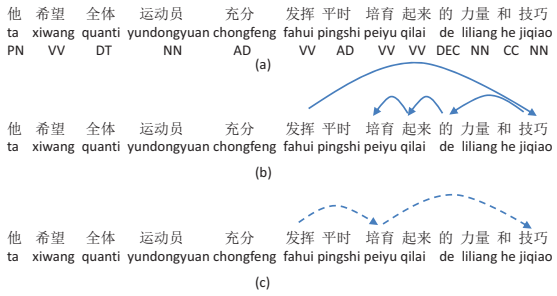


Figure 2: Example of an ambiguity on the Chinese side

In its English translation (Figure 3), word "that" is a clue indicating the relative clause which shows the relation between "skill" and "cultivate", as shown in Figure 3. The figure shows that the translation can provide useful bilingual constraints. From the dependency tree on the target side, we find that the word "skill" corresponding to "技巧(jiqiao)/skill" depends on the word "demonstrate" corresponding to "发挥(fahui)/demonstrate", while the word "cultivate" corresponding to "培育(peiyu)/cultivate" is a grandchild of "skill". This is a positive evidence for supporting "发挥(fahui)/demonstrate" as being the head of "技巧(jiqiao)/skill".

The above case uses the human translation on the target side. However, there are few human-annotated bilingual treebanks and the existing bilingual treebanks are usually small. In contrast, there are large-scale monolingual treebanks, e.g., the PTB and the latest version of CTB. So we want to use existing resources to generate a bilingual treebank with the help of SMT systems. We hope to improve source side parsing by using this newly built bilingual treebank.

Figure 3: Example of human translation

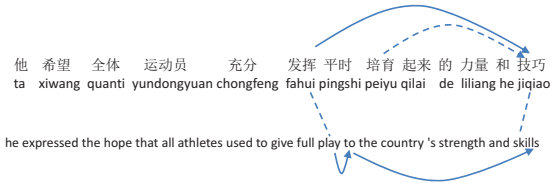


Figure 4: Example of Moses translation

Figure 4 shows an example of a translation using a Moses-based system, where the target sentence is parsed by a monolingual target parser. The translation contains some errors, but it does contain some correct parts that can be used for disambiguation. In the figure, the word "skills" corresponding to "技巧(jiqiao)/skill" is a grandchild of the word "play" corresponding to "发挥(fahui)/demonstrate". This is a positive evidence for supporting "发挥(fahui)/demonstrate" as being the head of "技巧(jiqiao)/skill".

From this example, although the sentences and parse trees on the target side are not perfect, we still can explore useful information to improve bitext parsing. In this paper, we focus on how to design a method to verify such unreliable bilingual constraints.

3 Parsing model

In this paper, we implement our approach based on graph-based parsing models (McDonald and Pereira, 2006; Carreras, 2007). Note that our approach can also be applied to transition-based parsing models (Nivre, 2003; Yamada and Matsumoto, 2003).

The graph-based parsing model is to search for the maximum spanning tree (MST) in a graph (McDonald and Pereira, 2006). The formulation defines the score of a dependency tree to be the sum of edge scores,

$$s(x, y) = \sum_{g \in y} score(w, x, g) = \sum_{g \in y} w \cdot f(x, g) \quad (1)$$

where x is an input sentence, y is a dependency tree for x, and g is a spanning subgraph of y. $f(x, g)$ can be based on arbitrary features of the subgraph and the input sequence x and the feature weight vector w are the parameters to be learned by using MIRA (Crammer and Singer, 2003) during training.

In our approach, we use two types of features for the parsing model. One is monolingual features based on the source sentences. The monolingual features include the first- and second- order features presented in McDonald and Pereira (2006) and the parent-child-grandchild features used in Carreras (2007). The other one is bilingual features (described in Sections 4 and 5) that consider the bilingual constraints.

We call the parser with the monolingual features on the source side Parser^s, and the parser with the monolingual features on the target side Parser^t.

4 Original bilingual features

In this paper, we generate two types of bilingual features, original and verified bilingual features. The original bilingual features (described in this section) are based on the bilingual constraints without being verified by large-scale unannotated data. And the verified bilingual features (described in Section 5) are based on the bilingual constraints verified by using large-scale unannotated data.

4.1 Auto-generated bilingual treebank

Assuming that we have monolingual treebanks on the source side, an SMT system that can translate the source sentences into the target language, and a Parser^t trained on the target monolingual treebank.

We first translate the sentences of the source monolingual treebank into the target language using the SMT system. Usually, SMT systems can output the word alignment links directly. If they can not, we perform word alignment using some publicly available tools, such as Giza++ (Och and Ney, 2003) or Berkeley Aligner (Liang et al., 2006; DeNero and Klein, 2007). The translated sentences are parsed by the Parser^t. Then, we have a newly auto-generated bilingual treebank.

4.2 Bilingual constraint functions

In this paper, we focus on the first- and second-order graph models (McDonald and Pereira, 2006; Carreras, 2007). Thus we produce the constraints for bigram (a single edge) and trigram (adjacent edges) dependencies in the graph model. For the trigram dependencies, we consider the parent-sibling and parent-child-grandchild structures described in McDonald and Pereira (2006) and Carreras (2007). We leave the third-order models (Koo and Collins, 2010) for a future study.

Suppose that we have a (candidate) dependency relation r_s that can be a bigram or trigram dependency. We examine whether the corresponding words of the source words of r_s have a dependency relation r_t in the target trees. We also consider the direction of the dependency relation. The corresponding word of the head should also be the head in r_t. We define a binary function for this bilingual constraint: $F_{bn}(r_{sn} : r_{tk})$, where n and k refers to the types of the dependencies (2 for bigram and 3 for trigram). For example, in $r_{s2} : r_{t3}$, r_{s2} is a bigram dependency on the source side and r_{t3} is a trigram dependency on the target side.

4.2.1 Bigram constraint function: F_{b2}

For r_{s2}, we consider two types of bilingual constraints. The first constraint function, denoted as $F_{b2}(r_{s2} : r_{t2})$, checks if the corresponding words also have a direct dependency relation r_{t2}. Figure 5 shows an example, where the source word "全体(quanti)" depends on "运动员(yundongyuan)" and word "all" corresponding to "全体(quanti)" depends on word "athletes" corresponding to "运动员(yundongyuan)". In this case, $F_{b2}(r_{s2} : r_{t2}) = +$. However, when the source words are "他(ta)" and "希望(xiwang)", this time their corresponding words "He" and "hope" do not have a direct dependency relation. In this case, $F_{b2}(r_{s2} : r_{t2}) = -$.

The second constraint function, denoted as $F_{b2}(r_{s2} : r_{t3})$, checks if the corresponding words form a parent-child-grandchild relation that often occurs in translation (Koehn et al., 2003). Figure 6 shows an example. The source word "技巧(jiqiao)" depends on "发挥(fahui)" while its corresponding word "skills" indirectly depends on "play" which corresponds to "发挥(fahui)" via "to". In this case, $F_{b2}(r_{s2} : r_{t3}) = +$.

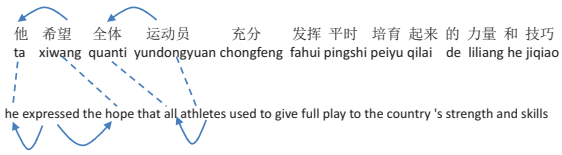


Figure 5: Example of bilingual constraints (2to2)

Figure 6: Example of bilingual constraints (2to3)

4.2.2 Trigram constraint function: F_{b3}

For a second-order relation on the source side, we consider one type of constraint. We have three source words that form a second-order relation and all of them have the corresponding words. We define function $F_{b3}(r_{s3} : r_{t3})$ for this constraint. The function checks if the corresponding words form a trigram dependencies structure. An example is shown in Figure 7. The source words "力量(liliang)", "和(he)", and "技巧(jiqiao)" form a parent-sibling structure, while their corresponding words "strength", "and", and "skills" also form a parent-sibling structure on the target side. In this case, function $F_{b3}(r_{s3} : r_{t3}) = +$.

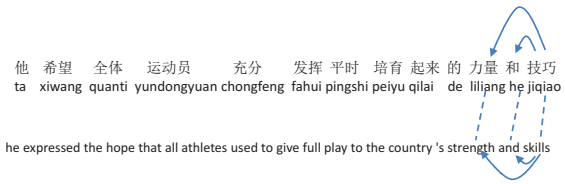


Figure 7: Example of bilingual constraints (3to3)

4.3 Bilingual reordering function: F_{ro}

Huang et al. (2009) proposed features based on reordering between languages for a shift-reduce parser. They define the features based on word-alignment information to verify whether the corresponding words form a contiguous span to resolve shift-reduce conflicts. We also implement similar features in our system. For example, in Figure 1-(a) the source span is [会谈(huitan), 结果(jieguo)], which maps onto [results, conference]. Because no word within this target span is aligned to a source word outside of the source span, this span is a contiguous span. In this case, function $F_{ro} = +$, otherwise $F_{ro} = -$.

4.4 Original bilingual features

We define original bilingual features based on the bilingual constraint functions and the bilingual reordering function.

Table 1 lists the original features, where Dir refers to the directions[1] of the source-side dependencies, F_{b2} can be $F_{b2}(r_{s2} : r_{t2})$ and $F_{b2}(r_{s2} : r_{t3})$, and F_{b3} is $F_{b3}(r_{s3} : r_{t3})$. Each line of the table defines a feature template that is a combination of functions.

First-order features	Second-order features
$\langle F_{ro} \rangle$	
$\langle F_{b2}, Dir \rangle$	$\langle F_{b3}, Dir \rangle$
$\langle F_{b2}, Dir, F_{ro} \rangle$	$\langle F_{b3}, Dir, F_{ro} \rangle$

Table 1: Original bilingual features

We use an example to show how to generate the original bilingual features in practice. In Figure 4, we want to define the bilingual features for the bigram dependency (r_{s2}) between "发挥(fahui)" and "技巧(jiqiao)". The corresponding words form a trigram relation r_{t3} in the target dependency tree. The direction of the bigram dependency is right. Then we have feature "$\langle F_{b2}(r_{s2} : r_{t3}) = +, RIGHT \rangle$" for the second first-order feature template in Table 1.

5 Verified bilingual features

However, because the bilingual treebank is generated automatically, using the bilingual constraints alone is not reliable. Therefore, in this section we verify the constraints by using large-scale unannotated data to overcome this problem. More specifically, r_{tk} of the constraint is verified by checking a list of target monolingual subtrees and $r_{sn} : r_{tk}$ is verified by checking a list of bilingual subtrees. The subtrees are extracted from the large-scale unannotated data. The basic idea is as follows: if the dependency structures of a bilingual constraint can be found in the list of the target monolingual subtrees

[1]For the second order features, Dir is the combination of the directions of two dependencies.

or bilingual subtrees, this constraint will probably be reliable.

We first parse the large-scale unannotated monolingual and bilingual data. Subsequently, we extract the monolingual and bilingual subtrees from the parsed data. We then verify the bilingual constraints using the extracted subtrees. Finally, we generate the bilingual features based on the verified results for the parsing models.

5.1 Verified constraint functions

5.1.1 Monolingual target subtrees

Chen et al. (2009) proposed a simple method to extract subtrees from large-scale monolingual data and used them as features to improve monolingual parsing. Following their method, we parse large unannotated data with the Parser^t and obtain the subtree list (ST_t) on the target side. We extract two types of subtrees: bigram (two words) subtree and trigram (three words) subtree.

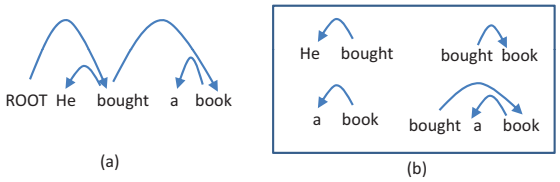


Figure 8: Example of monolingual subtree extraction

From the dependency tree in Figure 8-(a), we obtain the subtrees shown in Figure 8-(b) where the first three are bigram subtrees and the last one is a trigram subtree. After extraction, we obtain the subtree list ST_t that includes two sets, one for bigram subtrees, and the other one for trigram subtrees. We remove the subtrees occurring only once in the data. For each set, we assign labels to the extracted subtrees according to their frequencies by using the same method as that of Chen et al. (2009). If the frequency of a subtree is in the top 10% in the corresponding set, it is labeled HF. If the frequency is between the top 20% and 30%, it is labeled MF. We assign the label LF to the remaining subtrees. We use $Type(st_t)$ to refer to the label of a subtree, st_t.

5.1.2 Verified target constraint function: $F_{vt}(r_{tk})$

We use the extracted target subtrees to verify the r_{tk} of the bilingual constraints. In fact, r_{tk} is a candidate subtree. If the r_{tk} is included in ST_t, function $F_{vt}(r_{tk}) = Type(r_{tk})$, otherwise $F_{vt}(r_{tk}) = ZERO$. For example, in Figure 5 the bigram structure of "all" and "athletes" can form a bigram subtree that is included ST_t and its label is HF. In this case, $F_{vt}(r_{t2}) = HF$.

5.1.3 Bilingual subtrees

We extract bilingual subtrees from a bilingual corpus, which is parsed by the Parser^s and Parser^t on both sides. We extract three types of bilingual subtrees: bigram-bigram (st_{bi22}), bigram-trigram (st_{bi23}), and trigram-trigram (st_{bi33}) subtrees. For example, st_{bi22} consists of a bigram subtree on the source side and a bigram subtree on the target side.

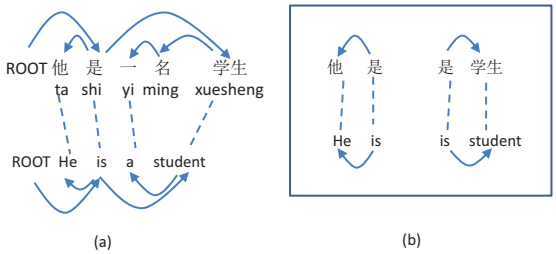


Figure 9: Example of bilingual subtree extraction

From the dependency tree in Figure 9-(a), we obtain the bilingual subtrees shown in Figure 9-(b). Figure 9-(b) shows the extracted bigram-bigram bilingual subtrees. After extraction, we obtain the bilingual subtrees ST_{bi}. We remove the subtrees occurring only once in the data.

5.1.4 Verified bilingual constraint function: $F_{vb}(r_{bink})$

We use the extracted bilingual subtrees to verify the r_{sn} : r_{tk} (r_{bink} in short) of the bilingual constraints. r_{sn} and r_{tk} form a candidate bilingual subtree st_{bink}. If the st_{bink} is included in ST_{bi}, function $F_{vb}(r_{bink}) = +$, otherwise $F_{vb}(r_{bink}) = -$.

5.2 Verified bilingual features

Then, we define another set of bilingual features by combining the verified constraint functions. We call these bilingual features 'verified bilingual features'.

Table 2 lists the verified bilingual features used in our experiments, where each line defines a feature template that is a combination of functions.

We use an example to show how to generate the verified bilingual features in practice. In Figure 4, we want to define the verified features for the bigram dependency (r_{s2}) between "发挥(fahui)" and "技巧(jiqiao)". The corresponding words form a trigram relation r_{t3}. The direction of the bigram dependency is right. Suppose we can find r_{t3} in ST_t with label MF and can not find the candidate bilingual subtree in ST_{bi}. Then we have feature "$\langle F_{b2}(r_{s2} : r_{t3}) = +, F_{vt}(r_{t3}) = MF, RIGHT\rangle$" for the third first-order feature template and feature "$\langle F_{b2}(r_{s2} : r_{t3}) = +, F_{vb}(r_{bi23}) = -, RIGHT\rangle$" for the fifth in Table 2.

First-order features	Second-order features
$\langle F_{ro}\rangle$	
$\langle F_{b2}, F_{vt}(r_{tk})\rangle$	$\langle F_{b3}, F_{vt}(r_{tk})\rangle$
$\langle F_{b2}, F_{vt}(r_{tk}), Dir\rangle$	$\langle F_{b3}, F_{vt}(r_{tk}), Dir\rangle$
$\langle F_{b2}, F_{vb}(r_{bink})\rangle$	$\langle F_{b3}, F_{vb}(r_{bink})\rangle$
$\langle F_{b2}, F_{vb}(r_{bink}), Dir\rangle$	$\langle F_{b3}, F_{vb}(r_{bink}), Dir\rangle$
$\langle F_{b2}, F_{ro}, F_{vb}(r_{bink})\rangle$	

Table 2: Verified bilingual features

6 Experiments

We evaluated the proposed method on the translated portion of the Chinese Treebank V2 (referred to as CTB2$_{tp}$) (Bies et al., 2007), articles 1-325 of CTB, which have English translations with gold-standard parse trees. The tool "Penn2Malt"[2] was used to convert the data into dependency structures. Following the studies of Burkett and Klein (2008), Huang et al. (2009) and Chen et al. (2010), we used the exact same data split: 1-270 for training, 301-325 for development, and 271-300 for testing. Note that we did not use human translation on the English side of this bilingual treebank to train our new parsers. For testing, we used two settings: a test with human translation and another with auto-translation. To process unannotated data, we trained a first-order Parser^s on the training data.

To prove that the proposed method can work on larger monolingual treebanks, we also tested our methods on the CTB7 (LDC2010T07) that includes much more sentences than CTB2$_{tp}$. We used articles 301-325 for development, 271-300 for testing, and the other articles for training. That is, we evaluated the systems on the same test data as CTB2$_{tp}$. Table 3 shows the statistical information on the data sets.

	Train	Dev	Test
CTB2$_{tp}$	2,745	273	290
CTB7	50,747	273	290

Table 3: Number of sentences of data sets used

We built Chinese-to-English SMT systems based on Moses[3]. Minimum error rate training (MERT) with respect to BLEU score was used to tune the decoder's parameters. The translation model was created from the FBIS corpus (LDC2003E14). We used SRILM[4] to train a 5-gram language model. The language model was trained on the target side of the FBIS corpus and the Xinhua news in English Gigaword corpus (LDC2009T13). The development and test sets were from NIST MT08 evaluation campaign[5]. We then used the SMT systems to translate the training data of CTB2$_{tp}$ and CTB7.

To directly compare with the results of Huang et al. (2009) and Chen et al. (2010), we also used the same word alignment tool, Berkeley Aligner (Liang et al., 2006; DeNero and Klein, 2007), to perform word alignment for CTB2$_{tp}$ and CTB7. We trained a Berkeley Aligner on the FBIS corpus (LDC2003E14). We removed notoriously bad links in {a, an, the}×{的(de), 了(le)} following the work of Huang et al. (2009).

To train an English parser, we used the PTB (Marcus et al., 1993) in our experiments and the tool "Penn2Malt" to convert the data. We split the data into a training set (sections 2-21), a development set (section 22), and a test set (section 23). We trained first-order and second-order Parser^t on the training data. The unlabeled attachment score (UAS) of the second-order Parser^t was 91.92, indicating state-of-the-art accuracy on the test data. We used the second-order Parser^t to parse the auto-translated/human-made target sentences in the CTB

[2] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

[3] http://www.statmt.org/moses/

[4] http://www.speech.sri.com/projects/srilm/download.html

[5] http://www.itl.nist.gov/iad/mig//tests/mt/2008/

data.

To extract English subtrees, we used the BLLIP corpus (Charniak et al., 2000) that contains about 43 million words of WSJ texts. We used the MX-POST tagger (Ratnaparkhi, 1996) trained on training data to assign POS tags and used the first-order Parser^t to process the sentences of the BLLIP corpus. To extract bilingual subtrees, we used the FBIS corpus and an additional bilingual corpus containing 800,000 sentence pairs from the training data of NIST MT08 evaluation campaign. On the Chinese side, we used the morphological analyzer described in (Kruengkrai et al., 2009) trained on the training data of CTB$_{tp}$ to perform word segmentation and POS tagging and used the first-order Parser^s to parse all the sentences in the data. On the English side, we used the same procedure as we did for the BLLIP corpus. Word alignment was performed using the Berkeley Aligner.

We reported the parser quality by the UAS, i.e., the percentage of tokens (excluding all punctuation tokens) with correct HEADs.

6.1 Experimental settings

For baseline systems, we used the monolingual features mentioned in Section 3. We called these features basic features. To compare the results of (Burkett and Klein, 2008; Huang et al., 2009; Chen et al., 2010), we used the test data with human translation in the following three experiments. The target sentences were parsed by using the second-order Parser^t. We used PAG to refer to our parsers trained on the auto-generated bilingual treebank.

6.2 Training with CTB2$_{tp}$

	Order-1	Order-2
Baseline	84.35	87.20
PAG$_o$	84.71(+0.36)	87.85(+0.65)
PAG	85.37(+1.02)	88.49(+1.29)
ORACLE	85.79(+1.44)	88.87(+1.67)

Table 4: Results of training with CTB2$_{tp}$

First, we conducted the experiments on the standard data set of CTB2$_{tp}$, which was also used in other studies (Burkett and Klein, 2008; Huang et al., 2009; Chen et al., 2010). The results are given in Table 4, where Baseline refers to the system with the basic features, PAG$_o$ refers to that after adding the original bilingual features of Table 1 to Baseline, PAG refers to that after adding the verified bilingual features of Table 2 to Baseline, and ORACLE[6] refers to using human-translation for training data with adding the features of Table 1. We obtained an absolute improvement of 1.02 points for the first-order model and 1.29 points for the second-order model by adding the verified bilingual features. The improvements of the final systems (PAG) over the Baselines were significant in McNemar's Test ($p < 0.001$ for the first-order model and $p < 0.0001$ for the second-order model). If we used the original bilingual features (PAG$_o$), the system dropped 0.66 points for the first-order and 0.64 points for the second-order compared with system PAG. This indicated that the verified bilingual constraints did provide useful information for the parsing models.

We also found that PAG was about 0.3 points lower than ORACLE. The reason is mainly due to the imperfect translations, although we used the large-scale subtree lists to help verify the constraints. We tried adding the features of Table 2 to the ORACLE system, but the results were worse. These facts indicated that our approach obtained the benefits from the verified constraints, while using the bilingual constraints alone was enough for ORACLE.

6.3 Training with CTB7

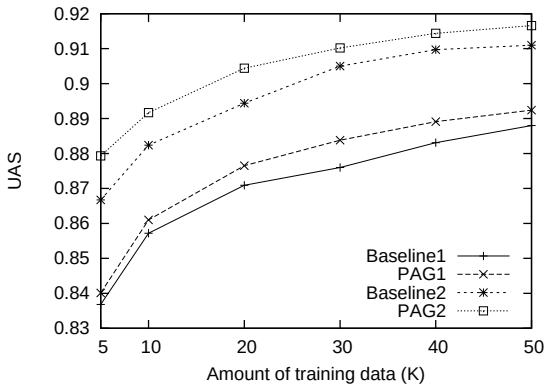


Figure 10: Results of using different sizes of training data

Here, we demonstrate that our approach is still able to provide improvement, even if we use larger

[6]Note that we also used the tool to perform the word alignment automatically.

	Baseline	D10	D20	D50	D100	GTran
BLEU	n/a	14.71	15.84	16.92	17.95	n/a
UAS	87.20	87.63	87.67	88.20	88.49	88.58

Table 5: Results of using different translations

training data that result in strong baseline systems. We incrementally increased the training sentences from the CTB7. Figure 10 shows the results of using different sizes of CTB7 training data, where the numbers of the x-axis refer to the sentence numbers of training data used, Baseline1 and Baseline2 refer to the first- and second-order baseline systems, and PAG1 and PAG2 refer to our first- and second-order systems. The figure indicated that our system always outperformed the baseline systems. For small data sizes, our system performed much better than the baselines. For example, when using 5,000 sentences, our second-order system provided a 1.26 points improvement over the second-order baseline. Finally, when we used all of the CTB7 training data, our system achieved 91.66 for the second-order model, while the baseline achieved 91.10.

6.4 With different settings of SMT systems

We investigated the effects of different settings of SMT systems. We randomly selected 10%, 20%, and 50% of FBIS to train the Moses systems and used them to translate CTB2$_{tp}$. The results are in Table 5, where D10, D20, D50, and D100 refer to the system with 10%, 20%, 50%, and 100% data respectively. For reference, we also used the Google-translate online system[7], indicated as GTran in the table, to translate the CTB2$_{tp}$.

From the table, we found that our system outperformed the Baseline even if we used only 10% of the FBIS corpus. The BLEU and UAS scores became higher, when we used more data of the FBIS corpus. And the gaps among the results of D50, D100, and GTran were small. This indicated that our approach was very robust to the noise produced by the SMT systems.

6.5 Testing with auto-translation

We also translated the test data into English using the Moses system and tested the parsers on the new test data. Table 6 shows the results. The results showed that PAG outperformed the baseline systems for both the first- and second-order models. This indicated that our approach can provide improvement in a purely monolingual setting with the help of SMT.

	Order-1	Order-2
Baseline	84.35	87.20
PAG	84.88(+0.53)	87.89(+0.69)

Table 6: Results of testing with auto-translation (training with CTB2$_{tp}$)

6.6 Comparison results

	With CTB2$_{tp}$		With CTB7	
Type	System	UAS	System	UAS
M	Baseline	87.20	Baseline	91.10
HA	Huang2009	86.3	n/a	
	Chen2010$_{BI}$	88.56		
	Chen2010$_{ALL}$	90.13		
AG	PAG	88.49	PAG	91.66
	PAG+ST$_s$	89.75		

Table 7: Comparison of our results with other previous reported systems. Type M denotes training on monolingual treebank. Types HA and AG denote training on human-annotated and auto-generated bilingual treebanks respectively.

We compared our results with the results reported previously for the same data. Table 7 lists the results, where Huang2009 refers to the result of Huang et al. (2009), Chen2010$_{BI}$ refers to the result of using bilingual features in Chen et al. (2010), and Chen2010$_{ALL}$ refers to the result of using all of the features in Chen et al. (2010). The results showed that our new parser achieved better accuracy than Huang2009 and comparable to Chen2010$_{BI}$. To achieve higher performance, we also added the source subtree features (Chen et al., 2009) to our system: PAG+ST$_s$. The new result is close to Chen2010$_{ALL}$. Compared with the approaches of

[7] http://translate.google.com/

Huang et al. (2009) and Chen et al. (2010), our approach used an auto-generated bilingual treebank while theirs used a human-annotated bilingual treebank. By using all of the training data of CTB7, we obtained a more powerful baseline that performed much better than the previous reported results. Our parser achieved 91.66, much higher accuracy than the others.

7 Conclusion

We have presented a simple yet effective approach to improve bitext parsing with the help of SMT systems. Although we trained our parser on an auto-generated bilingual treebank, we achieved an accuracy comparable to the systems trained on human-annotated bilingual treebanks on the standard test data. Moreover, our approach continued to provide improvement over the baseline systems when we used a much larger monolingual treebank (over 50,000 sentences) where target human translations are not available and very hard to construct. We also demonstrated that the proposed approach can be effective in a purely monolingual setting with the help of SMT.

Acknowledgments

This study was started when Wenliang Chen, Yujie Zhang, and Yoshimasa Tsuruoka were members of Language Infrastructure Group, National Institute of Information and Communications Technology (NICT), Japan. We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work.

References

Ann Bies, Martha Palmer, Justin Mott, and Colin Warner. 2007. English Chinese Translation Treebank V 1.0, LDC2007T02. *Linguistic Data Consortium*.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP 2008*, pages 877–886, Honolulu, Hawaii, October. Association for Computational Linguistics.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic, June. Association for Computational Linguistics.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC2000T43. *Linguistic Data Consortium*.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP 2009*, pages 570–579, Singapore, August.

Wenliang Chen, Jun'ichi Kazama, and Kentaro Torisawa. 2010. Bitext dependency parsing with bilingual subtree constraints. In *Proceedings of ACL 2010*, pages 21–29, Uppsala, Sweden, July. Association for Computational Linguistics.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of ACL 2007*, pages 17–24, Prague, Czech Republic, June. Association for Computational Linguistics.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of EMNLP 2009*, pages 1222–1231, Singapore, August. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL 2003*, pages 48–54. Association for Computational Linguistics.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL 2010*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging. In *Proceedings of ACL-IJCNLP2009*, pages 513–521, Suntec, Singapore, August. Association for Computational Linguistics.

Charles N. Li and Sandra A. Thompson. 1997. *Mandarin Chinese - A Functional Reference Grammar*. University of California Press.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of NAACL 2006*, pages 104–111, New York City, USA, June. Association for Computational Linguistics.

Yang Liu and Liang Huang. 2010. Tree-based and forest-based translation. In *Tutorial Abstracts of ACL 2010*, page 2, Uppsala, Sweden, July. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguisticss*, 19(2):313–330.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL 2006*, pages 81–88.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT2003*, pages 149–160.

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

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP 1996*, pages 133–142.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of EMNLP 2004*, pages 49–56.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT 2003*, pages 195–206.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of ACL-IJCNLP2009*, pages 55–63, Suntec, Singapore, August. Association for Computational Linguistics.

Accurate Parsing with Compact Tree-Substitution Grammars: Double-DOP

Federico Sangati and Willem Zuidema
Institute for Logic, Language and Computation
University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
{f.sangati,zuidema}@uva.nl

Abstract

We present a novel approach to Data-Oriented Parsing (DOP). Like other DOP models, our parser utilizes syntactic fragments of arbitrary size from a treebank to analyze new sentences, but, crucially, it uses only those which are encountered at least twice. This criterion allows us to work with a relatively small but representative set of fragments, which can be employed as the symbolic backbone of several probabilistic generative models. For parsing we define a transform-backtransform approach that allows us to use standard PCFG technology, making our results easily replicable. According to standard Parseval metrics, our best model is on par with many state-of-the-art parsers, while offering some complementary benefits: a simple generative probability model, and an explicit representation of the larger units of grammar.

1 Introduction

Data-oriented Parsing (DOP) is an approach to wide-coverage parsing based on assigning structures to new sentences using fragments of variable size extracted from a treebank. It was first proposed by Scha in 1990 and formalized by Bod (1992), and preceded many developments in statistical parsing (e.g., the "treebank grammars" of Charniak 1997) and linguistic theory (e.g., the current popularity of "constructions", Jackendoff 2002). A rich literature on DOP has emerged since, yielding state-of-the-art results on the Penn treebank benchmark test (Bod, 2001; Bansal and Klein, 2010) and inspiring developments in related frameworks including tree kernels (Collins and Duffy, 2002), reranking (Charniak and Johnson, 2005) and Bayesian adaptor and fragment grammars (e.g., Johnson et al., 2007; O'Donnell et al., 2009; Cohn et al., 2010). By formalizing the idea of using large fragments of earlier language experience to analyze new sentences, DOP captures an important property of language cognition that has shaped natural language. It therefore complements approaches that have focused on properties like lexicalization or incrementality, and might bring supplementary strengths in other NLP tasks.

Early versions of DOP (e.g., Bod et al., 2003) aimed at extracting all subtrees of all trees in the treebank. The total number of constructions, however, is prohibitively large for non-trivial treebanks: it grows exponentially with the length of the sentences, yielding the astronomically large number of approximately 10^{48} for section 2-21 of the Penn WSJ corpus. These models thus rely on a big sample of fragments, which inevitably includes a substantial portion of overspecialized constructions. Later DOP models have used the Goodman transformation (Goodman, 1996, 2003) to obtain a compact representation of all fragments in the treebank (Bod, 2003; Bansal and Klein, 2010). In this case the grammatical constructions are no longer explicitly represented, and substantial engineering effort is needed to optimally tune the models and make them efficient.

In this paper we present a novel DOP model (Double-DOP) in which we extract a restricted yet representative subset of fragments: those recurring at least twice in the treebank. The explicit representation of the fragments allows us to derive simple

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 84–95,
Edinburgh, Scotland, UK, July 27–31, 2011.

ways of estimating probabilistic models on top of the symbolic grammar. This and other implementation choices aim at making the methodology transparent and easily replicable. The accuracy of Double-DOP is well within the range of state-of-the-art parsers currently used in other NLP-tasks, while offering the additional benefits of a simple generative probability model and an explicit representation of grammatical constructions.

The contributions of this paper are summarized as follows: (i) we describe an efficient tree-kernel algorithm which allows us to extract all recurring fragments, reducing the set of potential elementary units from the astronomical 10^{48} to around 10^6. (ii) We implement and compare different DOP estimation techniques to induce a probability model (PTSG) on top of the extracted symbolic grammar. (iii) We present a simple transformation of the extracted fragments into CFG-rules that allows us to use off-the-shelf PCFG parsing and inference. (iv) We integrate Double-DOP with recent state-splitting approaches (Petrov et al., 2006), yielding an even more accurate parser and a better understanding of the relation between DOP and state-splitting.

The rest of the paper is structured as follows. In section 2 we describe the symbolic backbone of the grammar formalism that we will use for parsing. In section 3 we illustrate the probabilistic extension of the grammar, including our transformation of PTSGs to PCFGs that allows us to use a standard PCFG parser, and a different transform that allows us to use a standard implementation of the inside-outside algorithm. In section 4 we present the experimental setup and the results.

2 The symbolic backbone

The basic idea behind DOP is to allow arbitrarily large fragments from a treebank to be the elementary units of production of the grammar. Fragments can be combined through *substitution* to obtain the phrase-structure tree of a new sentence. Figure 1 shows an example of a complete syntactic tree obtained by combining three elementary fragments. As in previous work, two fragments f_i and f_j can be combined ($f_i \circ f_j$) only if the leftmost substitution site $X\downarrow$ in f_i has the same label as the root node of f_j; in this case the resulting tree will correspond to f_i with f_j replacing X. The DOP formalism is discussed in detail in e.g., Bod et al. (2003).

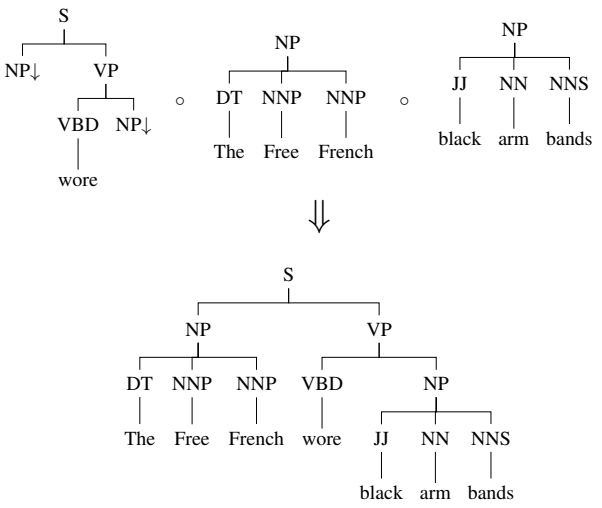


Figure 1: An example of a derivation of a complete syntactic structure (below) obtained combining three elementary fragments (above) by means of the substitution operation ∘. Substitution sites are marked with ↓.

2.1 Finding Recurring Fragments

The first step to build a DOP model is to define its symbolic grammar, i.e. the set of elementary fragments in the model. In the current work we explicitly extract a subset of fragments from the training treebank. To limit the fragment set size, we use a simple but heretofore unexplored constraint: we extract only those fragments that occur two or more times in the treebank[1]. Extracting this particular set of fragments is not trivial, though: a naive approach that filters a complete table of fragments together with their frequencies fails because that set, in a reasonably sized treebank, is astronomically large. Instead, we use a dynamic programming algorithm based on tree-kernel techniques (Collins and Duffy, 2001; Moschitti, 2006; Sangati et al., 2010).

The algorithm iterates over every pair of trees in

[1]More precisely we extract only the largest shared fragments for all pairs of trees in the treebank. All subtrees of these extracted fragments necessarily also occur at least twice, but they are only explicitly represented in our extracted set if they happen to form a largest shared fragment from another pair of trees. Hence, if a large tree occurs twice in the treebank the algorithm will extract from this pair only the full tree as a fragment and not all its (exponentially many) subtrees.

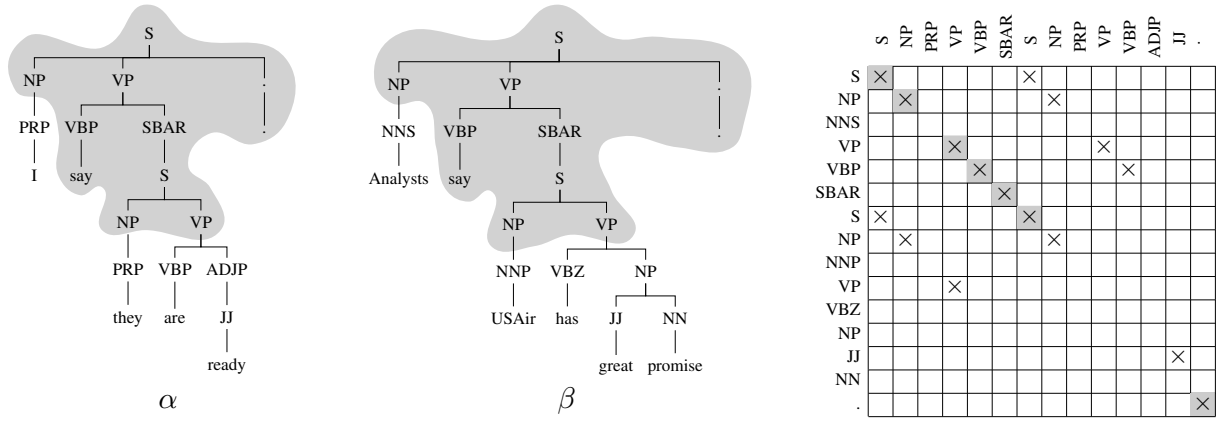


Figure 2: Left: example of two trees sharing a single maximum fragment, circled in the two trees. Right: the chart $\mathcal{M}$ which is used in the dynamic algorithm to extract all maximum fragments shared between the two trees. The highlighted cells in the chart are the ones which contribute to extract the shared fragment. The marked cells are those for which the corresponding nodes in the two tree have equivalent labels but differ in their lists of child nodes.

the treebank to look for common fragments. Figure 2 shows an example of a pair of trees $\langle \alpha, \beta \rangle$ being compared. The algorithm builds a chart $\mathcal{M}$ with one column for every indexed non-terminal node α_i in α, and one row for every indexed non-terminal node β_j in β. Each cell $\mathcal{M}\langle i, j \rangle$ identifies a set of indices corresponding to the largest fragment in common between the two trees starting from α_i and β_j. This set is empty if α_i and β_j differ in their labels, or they don't have the same list of child nodes. Otherwise (if both the labels and the lists of children match) the set is computed recursively as follows:

$$\mathcal{M}\langle i, j \rangle = \{\alpha_i\} \cup \left(\bigcup_{c=\{1,2,\ldots,|ch(\alpha)|\}} \mathcal{M}\langle ch(\alpha_i, c), ch(\beta_j, c) \rangle \right) \quad (1)$$

where $ch(\alpha)$ returns the indices of α's children, and $ch(\alpha, c)$ the index of its c^{th} child.

After filling the chart, the algorithm extracts the set of recurring fragments, and stores them in a table to keep track of their counts. This is done by converting back each fragment implicitly defined in every cell-set[2], and filtering out those that are properly contained in others.

In a second pass over the treebank, exact counts are obtained for each fragment in the extracted set. Parse trees in the training corpus are not necessarily covered entirely by recurring fragments; to ensure coverage, we also include in the symbolic backbone of our Double-DOP model all PCFG-productions not included in the set of extracted fragments.

2.2 Comparison with previous DOP work

Explicit grammars The number of recurring fragments in our symbolic grammar, extracted from the training sections of the Penn WSJ treebank[3], is around 1 million, and thus is significantly lower than previous work extracting explicit fragments (e.g., Bod, 2001, used more than 5 million fragments up to depth 14).

When looking at the extracted fragments we ask if we could have predicted which fragments occur twice or more. Figure 3 attempts to tackle this question by reporting some statistics on the extracted fragments. The majority of fragments are rather small with a limited number of words or substitution sites in the frontier. Yet, there is a significant portion of fragments, in the tail of the distribution, with more than 10 words or substitution sites. Since the space of all fragments with such characteristics is enormously large, selecting big recurring fragments using random sampling technique is like finding a needle in a haystack. Hence, random sampling processes (like Bod, 2001), will tend to represent fre-

[2] A cell-set containing a single index corresponds to the fragment including the node with that index together with all its children.

[3] This is after the treebank has been preprocessed. See also section 4.

quent recurring constructions such as *from NP to NP* or *whether S or not*, together with infrequent overspecialized fragments like *from Houston to NP*, while missing large generic constructions such as *everything you always wanted to know about NP but were afraid to ask*. These large constructions are excluded completely by models that only allow elementary trees up to a certain depth (typically 4 or 5) into the symbolic grammar (Zollmann and Sima'an, 2005; Zuidema, 2007; Borensztajn et al., 2009), or only elementary trees with exactly one lexical anchor (Sangati and Zuidema, 2009).

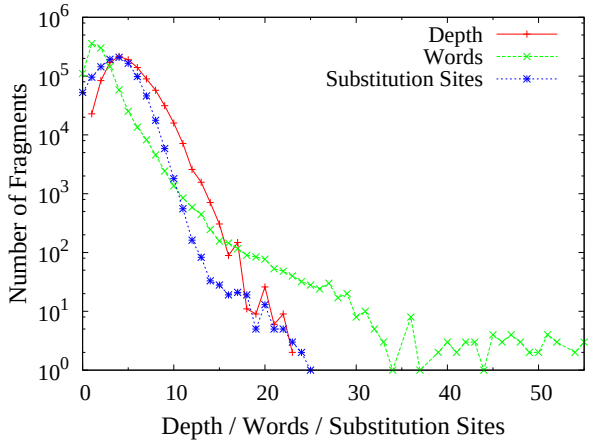


Figure 3: Distribution of the recurring fragments types according to several features: depth, number of words, and number of substitution sites. Their corresponding curves peak at 4 (depth), 1 (words), and 4 (substitution sites).

Implicit grammars Goodman (1996, 2003) defined a transformation for some versions of DOP to an equivalent PCFG-based model, with the number of rules extracted from each parse tree linear in the size of the trees. This transform, representing larger fragments only *implicitly*, is used in most recent DOP parsers (e.g., Bod, 2003; Bansal and Klein, 2010). Bod has promoted the Goodman transform as the solution to the computational challenges of DOP (e.g., Bod, 2003); it's important to realize, however, that the resulting grammars are still very large: WSJ sections 2-21 yield about 2.5 million rules in the basic version of Goodman's transform. Moreover, the transformed grammars differ from untransformed DOP grammars in that larger fragments are no longer explicitly represented. Rather, information about their frequency is distributed over many CFG-rules: if a construction occurs n times and contains m context-free productions, Goodman's transform uses the weights of $7nm + m$ rules to encode this fact. Thus, the information that the idiomatic fragment *(PP (IN "out") (PP (IN "of") (NP (NN "town")))))* occurs 3 times in WSJ sections 2-21, is distributed over 132 rules. This way, an attractive feature of DOP, viz. the explicit representation of the 'productive units' of language, is lost[4].

In addition, grammars that implicitly encode all fragments found in a treebank are strongly biased to over-represent big constructions: the great majority of the entire set of fragments belongs in fact to the largest tree in the treebank[5]. DOP models relying on Goodman's transform, need therefore to counteract this tendency. Bansal and Klein (2010), for instance, rely on a sophisticated tuning technique to correctly adjust the weights of the rules in the grammar. In our Double-DOP approach, instead, the number of fragments extracted from each tree varies much less (it ranges between 4 and 1,759). This comparison is shown in figure 4.

3 The probabilistic model

Like CFG grammars, our symbolic model produces extremely many parse trees for a given test sentence. We therefore need to disambiguate between the possible parses by means of a probability model that assigns probabilities to fragments, and defines a proper distribution over the set of possible full parse trees. For every nonterminal X in the treebank we have:

$$\sum_{f \in F_X} p(f) = 1 \qquad (2)$$

where F_X is the set of fragments in our symbolic grammar rooted in X. A derivation $d = f_1, f_2, \ldots, f_n$ of t is a sequence of the fragments that through left-most substitution produces t. The probability of a derivation is computed as the product of

[4]Bansal and Klein (2010) address this issue for contiguous constructions by extending the Goodman transform with a 'Packed Graph Encoding' for fragments that "bottom out in terminals". However, constructions with variable slots, such as *whether S or not*, are left unchanged.

[5]In fact, the number of extracted fragments increase exponentially with the size of the tree.

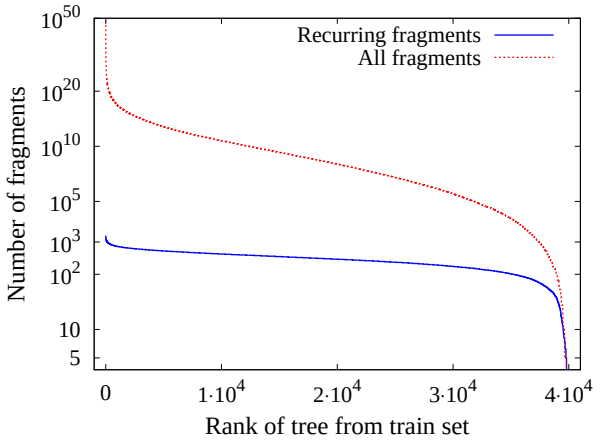


Figure 4: Number of fragments extracted from each tree in sections 2-21 of the WSJ treebank, when considering all-fragments (dotted line) and recurring-fragments (solid line). Trees on the x-axis are ranked according to the number of fragments. Note the double logarithmic scale on the y-axis.

the probability of each of its fragments.

$$P(d) = \prod_{f \in d} p(f) \quad (3)$$

In section 3.2 we describe ways of obtaining different probability distributions over the fragments in our grammar. In the following section we assume a given probabilistic model, and illustrate how to use standard PCFG parsing.

3.1 Parsing

It is possible to define a simple transform of our probabilistic fragment grammar, such that off-the-shelf parsers can be used. In order to perform the PTSG/PCFG conversion, every fragment in our grammar must be mapped to a CFG rule which will keep the same probability as the original fragment. The corresponding rule will have as the left hand side the root of the fragment and as the right hand side its yield, i.e., a sequence of terminals and non-terminals (substitution sites).

It might occur that several fragments are mapped to the same CFG rule[6]. These are interesting cases of syntactic ambiguity as shown in figure 5. In order to resolve this problem we need to map each ambiguous fragment to two unique CFG rules chained by a unique artificial node, as shown at the bottom of the same figure. To the first CFG rule in the chain we assign the probability of the fragment, while the second will receive probability 1, so the product gives back the original probability. The ambiguous and unambiguous PTSG/PCFG mappings need to be stored in a table, in order to convert back the compressed CFG derivations to the original PTSG model after parsing.

Such a transformed PCFG will generate the same derivations as the original PTSG grammar with identical probabilities. In our experiment we use a standard PCFG parser to produce a list of k-best Viterbi derivations. These, in turn, will be used to maximize possible objectives as described in section 3.3.

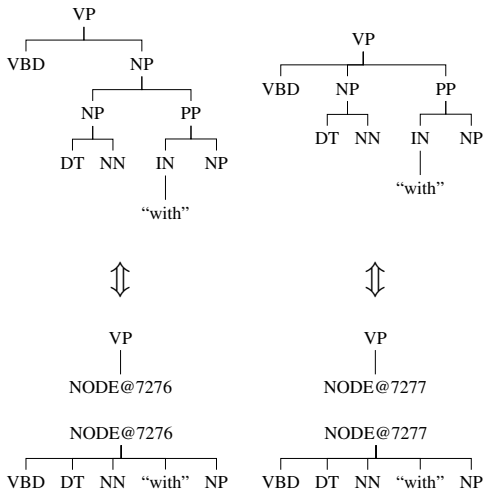


Figure 5: Above: example of 2 ambiguous fragments mapping to the same CFG rule VP → VBD DT NN "with" NP. The first fragment occurs 5 times in the training treebank, (e.g. in the sentence *was an executive with a manufacturing concern*) while the second fragment occurs 4 times (e.g. in the sentence *began this campaign with such high hopes*). Below: the two pairs of CFG rules that are used to map the two fragments to separate CFG derivations.

3.2 Inducing probability distributions

Relative Frequency Estimate (RFE) The simplest way to assign probabilities to fragments is to make them proportional to their counts[7] in the training set. When enforcing equation 2, that gives the

[6] In our binarized treebank we have 31,465 fragments types that are ambiguous in this sense.

[7] We refer to the counts of each fragment as returned by our extraction algorithm in section 2.1.

Relative Frequency Estimate (RFE):

$$p_{\text{RFE}}(f) = \frac{count(f)}{\sum_{f' \in F_{root(f)}} count(f')} \quad (4)$$

Unlike RFE for PCFGs, however, the RFE for PTSGs has no clear probabilistic interpretation. In particular, it does not yield the maximum likelihood solution, and when used as an estimator for an all-fragments grammar, it is strongly biased since it assigns the great majority of the probability mass to big fragments (Johnson, 2002). As illustrated in figure 4 this bias is much weaker when restricting the set of fragments with our approach. Although this does not solve all theoretical issues, it makes RFE a reasonable first choice again.

Equal Weights Estimate (EWE) Various other ways of choosing the weights of a DOP grammar have been worked out. The best empirical results have been reported by Bod (2003) with the EWE proposed by Goodman (2003). Goodman defined it for grammars in the Goodman transform, but for explicit grammars it becomes:

$$w_{\text{EWE}}(f) = \sum_{t \in TB} \frac{count(f,t)}{|\{f' \in t\}|} \quad (5)$$

$$p_{\text{EWE}}(f) = \frac{w_{\text{EWE}}(f)}{\sum_{f' \in F_{root(f)}} w_{\text{EWE}}(f')} \quad (6)$$

where the first sum is over all parse trees t in the treebank (TB), $count(f,t)$ gives the number of times fragment f occurs in t, and $|\{f' \in t\}|$ is the total number of subtrees of t that were included in the symbolic grammar.

Maximum Likelihood (ML) For reestimation, we can aim at maximizing the likelihood (ML) of the treebank. For this, it turns out that we can define another transformation of our PTSG, such that we can apply standard Inside-Outside algorithm for PCFGs (Lari and Young, 1990). The original version of IO is defined over *string rewriting* PCFGs, and maximizes the likelihood of the training set consisting of plain sentences. Reestimation shifts probability mass between alternative parse trees for a sentence. In contrast, our grammars consist of fragments of various size, and our training set consists of parse trees. Reestimation here shifts probability mass between alternative derivations for a parse tree.

Our transformation approach is illustrated with an example in figure 6. In step (b) the fragments in the grammar as well as the original parse trees in the treebank are "flattened" into bracket notation. In step (c) each fragment is transformed into a CFG rule in the transformed meta-grammar, whose right-hand side is constituted by the bracket notation of the fragment. Each substitution site $X\!\downarrow$ is raised to a meta-nonterminal X', and all other symbols, including parentheses, become meta-terminals. The left-hand side of the rule is constituted by the original root symbol R of the fragment raised to a meta-nonterminal R'.

The resulting PCFG generates trees in bracket notation, and we can run an of-the-shelf inside-outside algorithm by presenting it parse trees from the train corpus in bracket notation[8]. In the experiments that we report below we used the RFE from section 3, to generate the initial weights for the grammar.

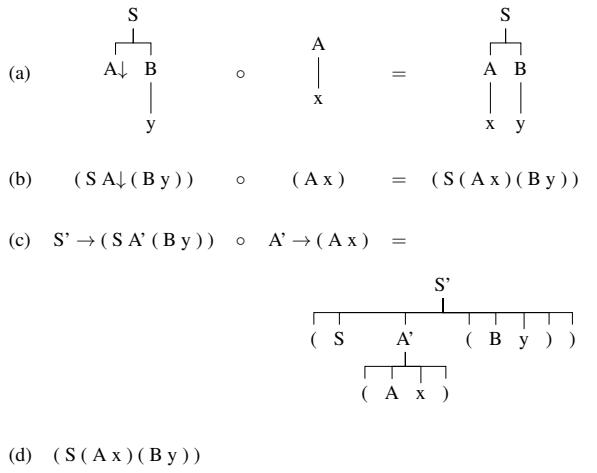


Figure 6: Rule and tree transforms that turn PTSG reestimation into PCFG reestimation; (a) a derivation of the sentence $x\,y$ through successive substitutions of elementary trees from a PTSG; (b) the same elementary trees and resulting parse tree in bracket notation; (c) an equivalent derivation with the meta-grammar, where the original substitution sites reappear as meta-nonterminals (marked with a prime) and all other symbols as meta-terminals; (d) the yield of the derivation in c.

[8] However, the results with inside-outside reported in this paper were obtained with an earlier version of our code that uses an equivalent but special-purpose implementation.

3.3 Maximizing Objectives

MPD The easiest objective in parsing, is to select the most probable derivation (MPD), obtained by maximizing equation 3.

MPP A DOP grammar can often generate the same parse tree t through different derivations $D(t) = d_1, d_2, \ldots d_m$. The probability of t is therefore obtained by summing the probabilities of all its possible derivations.

$$P(t) = \sum_{d \in D(t)} p(d) = \sum_{d \in D(t)} \prod_{f \in d} p(f) \quad (7)$$

An intuitive objective for a parser is to select, for a given sentence, the parse tree with highest probability according to equation 7, i.e., the most probable parse (MPP): unfortunately, identifying the MPP is computationally intractable (Sima'an, 1996). However, we can approximate the MPP by deriving a list of k-best derivations, summing up the probabilities of those resulting in the same parse tree, and select the tree with maximum probability.

MCP, MRS Following Goodman (1998), Sima'an (1999, 2003), and others, we also consider other objectives, in particular, the max constituent parse (MCP), and the max rule sum (MRS).

MCP maximizes a weighted average of the expected labeled recall L/N_C and (approximated) labeled precision L/N_G under the given posterior distribution, where L is the number of correctly labeled constituents, N_C the number of constituents in the correct tree, and N_G the number of constituents in the guessed tree. Recall is easy to maximize since the estimated N_C is constant. L/N_C can be in fact maximized in:

$$\hat{t} = \arg\max_t \sum_{lc \in t} P(lc) \quad (8)$$

where lc ranges over all labeled constituents in t and $P(lc)$ is the marginalized probability of all the derivation trees in the grammar yielding the sentence under consideration which contains lc.

Precision, instead, is harder because the denominator N_G depends on the chosen guessed tree. Goodman (1998) proposes to look at another metric which is strongly correlated with precision, which is the mistake rate $(N_G - L)/N_C$ that we want to minimize. We combine recall with mistake rate through linear interpolation:

$$\hat{t} = \arg\max_t \mathcal{E}(\frac{L}{N_C} - \lambda \frac{N_G - L}{N_C}) \quad (9)$$

$$= \arg\max_t \sum_{lc \in t} P(lc) - \lambda(1 - P(lc)) \quad (10)$$

where 10 is obtained from 9 assuming N_C constant, and the optimal level for λ has to be evaluated empirically.

Unlike MPP, the MCP can be calculated efficiently using dynamic programming techniques over the parse forest. However, in line with the aims of this paper to produce an easily reproducible implementation of DOP, we developed an accurate approximation of the MCP using a list of k-best derivations, such as those that can be obtained with an off-the-shelf PCFG parser.

We do so by building a standard CYK chart, where every cell corresponds to a specific span in the test sentence. We store in each cell the probability of seeing every label in the grammar yielding the corresponding span, by marginalizing the probabilities of all the parse trees in the obtained k-best derivations that contains that label covering the same span. We then compute the Viterbi-best parse maximizing equation 10.

We implement max rule sum (MRS) in a similar way, but do not only keep track of labels in every cell, but of each CFG rule that span the specific yield (see also Sima'an, 1999, 2003). We haven't implemented the max rule product (MRP) where posteriors are multiplied instead of added (Petrov and Klein, 2007; Bansal and Klein, 2010).

4 Experimental Setup

In order to build and test our Double-DOP model[9], we employ the Penn WSJ Treebank (Marcus et al., 1993). We use sections 2-21 for training, section 24 for development and section 23 for testing.

Treebank binarization We start with some preprocessing of the treebank, following standard prac-

[9]The software produced for running our model is publicly available and included in the supplementary material to this paper. To the best of our knowledge this is the first DOP software released that can be used to parse the WSJ PTB.

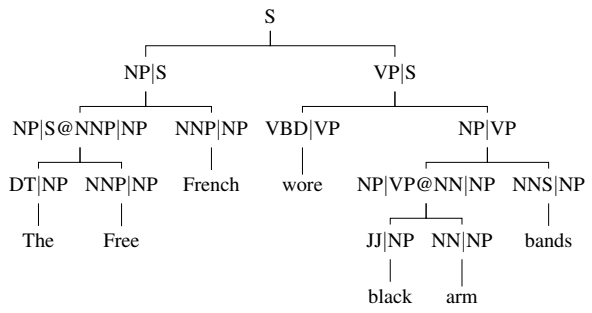


Figure 7: The binarized version of the tree in figure 1, with H=1 and P=1.

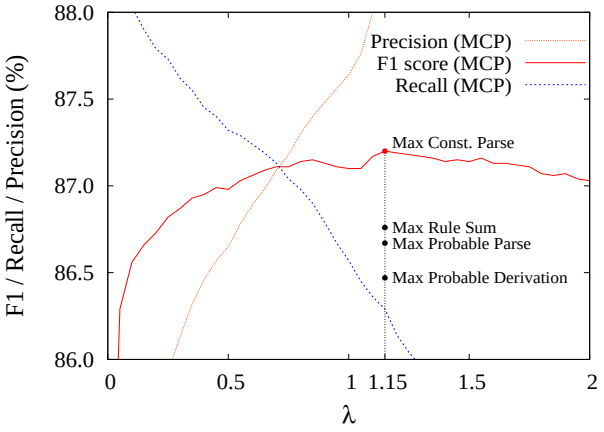


Figure 8: Double-DOP results on the development section (≤ 40) with different maximizing objectives.

tice in WSJ parsing. We remove traces and functional tags. We apply a left binarization of the training treebank as in Matsuzaki et al. (2005) and Klein and Manning (2003), setting the horizontal history H=1 and the parent labeling P=1. This means that when a node has more than 2 children, the i^{th} child (for $i \geq 3$) is conditioned on child $i-1$. Moreover the labels of all non-lexical nodes are enriched with the labels of their parent node. Figure 7 shows the binarized version of the tree structure in figure 1.

Unknown words We replace words appearing less than 5 times in the training data by one of 50 unknown word categories based on the presence of lexical features as implemented in Petrov (2009). In some of the experiments we also perform a smoothing over the lexical elements assigning low counts ($\epsilon = 0.01$) to open-class $\langle$words, PoS-tags$\rangle$ pairs not encountered in the training corpus[10].

Fragment extraction We extract the symbolic grammar and fragment frequencies from this preprocessed treebank as explained in section 2. This is the the most time-consuming step (around 160 CPU hours[11]).

In the extracted grammar we have in total 1,029,342 recurring fragments and 17,768 unseen CFG rules. We test several probability distributions over the fragments (section 3.2) and various maximization objectives (section 3.3).

[10] A PoS-tag is an open class if it rewrites to at least 50 different words in the training corpus. A word is an open class word if it has been seen only with open-class PoS-tags.

[11] Although our code could still be optimized further, it does already allow for running the job on M CPUs in parallel, reducing the time required by a factor M (10 hours with 16-CPUs).

Parsing We convert our PTSG into a PCFG (section 3.1) and use `Bitpar`[12] for parsing. For approximating MPP and other objectives we marginalize probabilities from the 1,000 best derivations.

4.1 Results

We start by presenting in figure 8 the results we obtain on the development set (section 24). Here we compare the *maximizing objectives* presented in section 3.3, using RFE to obtain the probability distribution over the fragments. We conclude that, empirically, MCP for $\lambda = 1.15$, is the best choice to maximize F1, followed by MRS, MPP, and MPD.

We also compare the various *estimators* presented in section 3.2, on the same development set, keeping MCP with $\lambda = 1.15$ as the maximizing objective. We find that RFE is the best estimator (87.2 F1[13]) followed by EWE (86.8) and ML (86.6). Our best results with ML are obtained when removing fragments occurring less than 6 times (apart from CFG-rules) and when stopping at the second iteration. This filtering is done in order to limit the number of big fragments in the grammar. It is well known that IO for DOP tends to assign most of the probability mass to big fragments, quickly overfitting the training data. It is surprising that EWE and ML perform worse than RFE, in contrast to earlier findings (Bod, 2003).

[12] `http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/BitPar.html`

[13] We computed F1 scores with EvalB (`http://nlp.cs.nyu.edu/evalb/`) using parameter file *new.prm*.

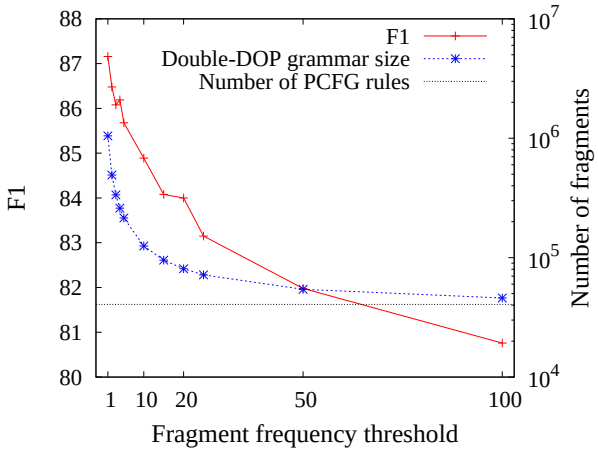


Figure 9: Performance (on the development set) and size of Double-DOP when considering only fragments whose occurring frequency in the training treebank is above a specific threshold (x-axis). In all cases, all PCFG-rules are included in the grammars. For instance, at the right-hand side of the plot a grammar is evaluated which included only 6754 fragments with a frequency > 100 as well as 39227 PCFG rules.

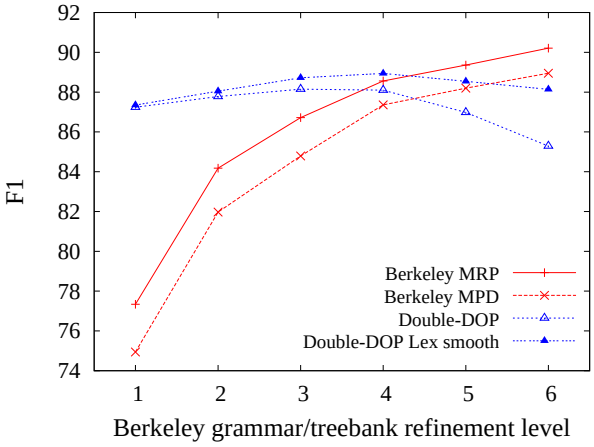


Figure 10: Comparison on section 24 between the performance of Double-DOP (using RFE and MCP with $\lambda = 1.15$, H=0, P=0) and Berkeley parser on different stages of refinement of the treebank/grammar.

We also investigate how a further restriction on the set of extracted fragments influences the performance of our model. In figure 9 we illustrate the performance of Double-DOP when restricting the grammar to fragments having frequencies greater than $1, 2, \ldots, 100$. We can notice a rather sharp decrease in performance as the grammar becomes more and more compact.

Next, we present some results on various Double-DOP grammars extracted from the same training treebank after refining it using the Berkeley state-splitting model[14] (Petrov et al., 2006; Petrov and Klein, 2007). In total we have 6 increasingly refined versions of the treebank, corresponding to the 6 cycles of the Berkeley model. We observe in figure 10 that our grammar is able to benefit from the state splits for the first four levels of refinement, reaching the maximum score at cycle 4, where we improve over our base model. For the last two data points, the treebank gets too refined, and using Double-DOP model on top of it, no longer improves accuracy.

We have also compared our best Double-DOP base model and the Berkeley parser on per-category performance. Here we observe an interesting trend: the Berkeley parser outperforms Double-DOP on very frequent categories, while Double-DOP performs better on infrequent ones. A detailed comparison is included in table 1.

Finally, in table 2 we present our results on the test set (section 23). Our best model (according to the best settings on the development set) performs slightly worse than the one by Bansal and Klein (2010) when trained on the original corpus, but outperforms it (and the version of their model with additional refinements) when trained on the refined version, in particular for the exact match score.

5 Conclusions

We have described Double-DOP, a novel DOP approach for parsing, which uses all constructions recurring at least twice in a treebank. This methodology is driven by the linguistic intuition that constructions included in the grammar should prove to be reusable in a representative corpus.

The extracted set of fragments is significantly smaller than in previous approaches. Moreover constructions are explicitly represented, which makes them potentially good candidates as semantic or translation units to be used in other applications.

Despite earlier reported excellent results with DOP parsers, they are almost never used in other

[14] We use the Berkeley grammar labeler following the base settings for the WSJ: trees are right-binarized, H=0, and P=0. Berkeley parser package is available at `http://code.google.com/p/berkeleyparser/`

Category label	% in gold	F1 Berkeley	F1 Double-DOP
NP	41.42	**91.4**	89.5
VP	20.46	**90.6**	88.6
S	13.38	**90.7**	87.6
PP	12.82	**85.5**	84.1
SBAR	3.47	**86.0**	82.1
ADVP	3.36	**82.4**	81.0
ADJP	2.32	**68.0**	67.3
QP	0.98	82.8	**84.6**
WHNP	0.88	**94.5**	92.0
WHADVP	0.33	**92.8**	91.9
PRN	0.32	**83.0**	77.9
NX	0.29	**9.50**	7.70
SINV	0.28	**90.3**	88.1
SQ	0.14	**82.1**	79.3
FRAG	0.10	26.4	**34.3**
SBARQ	0.09	84.2	**88.2**
X	0.06	72.0	**83.3**
NAC	0.06	54.6	**88.0**
WHPP	0.06	**91.7**	44.4
CONJP	0.04	55.6	**66.7**
LST	0.03	**61.5**	33.3
UCP	0.03	30.8	**50.0**
INTJ	0.02	44.4	**57.1**

Table 1: Comparison of the performance (per-category F1 score) on the development set between the Berkeley parser and the best Double-DOP model.

Parsing Model	test (≤ 40) F1	test (≤ 40) EX	test (all) F1	test (all) EX
PCFG Baseline				
PCFG (H=1, P=1)	77.6	17.2	76.5	15.9
PCFG (H=1, P=1) Lex smooth.	78.5	17.2	77.4	16.0
FRAGMENT-BASED PARSERS				
Zuidema (2007)*	83.8	26.9	-	-
Cohn et al. (2010) MRS	85.4	27.2	84.7	25.8
Post and Gildea (2009)	82.6	-	-	-
Bansal and Klein (2010) MCP	88.5	33.0	87.6	30.8
Bansal and Klein (2010) MCP + Additional Refinement	88.7	33.8	88.1	31.7
THIS PAPER				
Double-DOP	87.7	33.1	86.8	31.0
Double-DOP Lex smooth.	**87.9**	**33.7**	**87.0**	**31.5**
Double-DOP-Sp	88.8	35.9	88.2	33.8
Double-DOP-Sp Lex smooth.	**89.7**	**38.3**	**89.1**	**36.1**
REFINEMENT-BASED PARSERS				
Collins (1999)	88.6	-	88.2	-
Petrov and Klein (2007)	**90.6**	**39.1**	**90.1**	**37.1**

Table 2: Summary of the results of different parsers on the test set (sec 23). Double-DOP experiments use RFE, MCP with $\lambda = 1.15$, H=1, P=1; those on state-splitting (Double-DOP-Sp) use Berkeley cycle 4, H=0, P=0. Results from Petrov and Klein (2007) already include smoothing which is performed similarly to our smoothing technique (see section 4). (* Results on a development set, with sentences up to length 20.)

NLP tasks: where other successful parsers often feature as components of machine translation, semantic role labeling, question-answering or speech recognition systems, DOP is conspicuously absent in these neighboring fields (but for a possible application of closely related formalisms see, e.g., Yamangil and Shieber, 2010). The reasons for this are many, but most important are probably the computational inefficiency of many instances of the approach, the lack of downloadable software and the difficulties with replicating some of the key results.

In this paper we have addressed all three obstacles: our efficient algorithm for identifying the recurrent fragments in a treebank runs in polynomial time. The transformation to PCFGs that we define allows us to use a standard PCFG parser, while retaining the benefit of explicitly representing larger fragments. A different transform also allows us to run the popular inside-outside algorithm. Although IO results are slightly worse than with the naive relative frequency estimate, it is important to establish that the standard method for dealing with latent information (i.e., the derivations of a given parse) is not the best choice in this case. We expect that other re-estimation methods, for instance Variational Bayesian techniques, could be formulated in the same manner.

Finally, the availability of our programs, as well as the third party software that we use, also addresses the replicability issue. Where some researchers in the field have been skeptical of the DOP approach to parsing, we believe that our independent development of a DOP parser adds credibility to the idea that an approach that uses very many large subtrees, can lead to very accurate parsers.

Acknowledgments

We gratefully acknowledge funding by the Netherlands Organization for Scientific Research (NWO): FS is funded through a Vici-grant "Integrating Cognition" (277.70.006) to Rens Bod, and WZ through a Veni-grant "Discovering Grammar" (639.021.612). We also thank Rens Bod, Gideon Borensztajn, Jos de Bruin, Andreas van Cranenburgh, Phong Le, Remko Scha, Khalil Sima'an and the anonymous reviewers for very useful comments.

References

Mohit Bansal and Dan Klein. 2010. Simple, accurate parsing with an all-fragments grammar. In *Proceedings of the 48th Annual Meeting of the ACL*, pages 1098–1107. Association for Computational Linguistics, Uppsala, Sweden.

Rens Bod. 1992. A computational model of language performance: Data oriented parsing. In *Proceedings COLING'92 (Nantes, France)*, pages 855–859. Association for Computational Linguistics, Morristown, NJ.

Rens Bod. 2001. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of the ACL*. Morgan Kaufmann, San Francisco, CA.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 19–26. Association for Computational Linguistics, Morristown, NJ, USA.

Rens Bod, Khalil Sima'an, and Remko Scha. 2003. *Data-Oriented Parsing*. University of Chicago Press, Chicago, IL, USA.

Gideon Borensztajn, Willem Zuidema, and Rens Bod. 2009. Children's Grammars Grow More Abstract with Age—Evidence from an Automatic Procedure for Identifying the Productive Units of Language. *Topics in Cognitive Science*, 1(1):175–188.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603. AAAI Press/MIT Press.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. 43nd Meeting of Association for Computational Linguistics (ACL 2005)*.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, 11:3053–3096.

Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 625–632. MIT Press.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of 40th Annual Meeting of the ACL*, pages 263–270. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA.

Michael J. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Joshua Goodman. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 143–152.

Joshua Goodman. 2003. Efficient parsing of DOP with PCFG-reductions. In Bod et al. (2003).

Joshua T. Goodman. 1998. *Parsing inside-out*. Ph.D. thesis, Harvard University, Cambridge, MA, USA.

Ray Jackendoff. 2002. *Foundations of Language*. Oxford University Press, Oxford, UK.

Mark Johnson. 2002. The dop estimation method is biased and inconsistent. *Computational Linguistics*, 28:71–76.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in Neural Information Processing Systems*, volume 16, pages 641–648.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting on ACL*, pages 423–430. Association for Computational Linguistics, Morristown, NJ, USA.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *ACL '05: Proceedings of the 43rd Annual Meeting on ACL*, pages 75–82. Association for Computational Linguistics, Morristown, NJ, USA.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *ECML*, pages 318–329. Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Proceedings, Berlin, Germany.

Timothy J. O'Donnell, Noah D. Goodman, and Joshua B. Tenenbaum. 2009. Fragment Grammars: Exploring Computation and Reuse in Language. Technical Report MIT-CSAIL-TR-2009-013, MIT.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 433–440. Association for Computational Linguistics, Morristown, NJ, USA.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the ACL; Proceedings of the Main Conference*, pages 404–411. Association for Computational Linguistics, Rochester, New York.

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48. Association for Computational Linguistics, Suntec, Singapore.

Federico Sangati and Willem Zuidema. 2009. Unsupervised Methods for Head Assignments. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 701–709. Association for Computational Linguistics, Athens, Greece.

Federico Sangati, Willem Zuidema, and Rens Bod. 2010. Efficiently extract recurring tree fragments from large treebanks. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), Valletta, Malta.

Remko Scha. 1990. Taaltheorie en taaltechnologie: competence en performance. In Q. A. M. de Kort and G. L. J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, LVVN-jaarboek, pages 7–22. Landelijke Vereniging van Neerlandici, Almere. [Language theory and language technology: Competence and Performance] in Dutch.

Khalil Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proceedings of the 16th conference on Computational linguistics*, pages 1175–1180. Association for Computational Linguistics, Morristown, NJ, USA.

Khalil Sima'an. 1999. *Learning Efficient Disambiguation*. Ph.D. thesis, Utrecht University and University of Amsterdam.

Khalil Sima'an. 2003. On maximizing metrics for syntactic disambiguation. In *Proceedings of the International Workshop on Parsing Technologies (IWPT'03)*.

Elif Yamangil and Stuart M. Shieber. 2010. Bayesian synchronous tree-substitution grammar induction and its application to sentence compression. In *Proceedings of the 48th Annual Meeting of the ACL*, ACL '10, pages 937–947. Association for Computational Linguistics, Stroudsburg, PA, USA.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2/3):367–388.

Willem Zuidema. 2007. Parsimonious Data-Oriented Parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 551–560. Association for Computational Linguistics, Prague, Czech Republic.

A Generate and Rank Approach to Sentence Paraphrasing

Prodromos Malakasiotis[*] and **Ion Androutsopoulos**[*+]
[*]Department of Informatics, Athens University of Economics and Business, Greece
[+]Digital Curation Unit – IMIS, Research Centre "Athena", Greece

Abstract

We present a method that paraphrases a given sentence by first generating candidate paraphrases and then ranking (or classifying) them. The candidates are generated by applying existing paraphrasing rules extracted from parallel corpora. The ranking component considers not only the overall quality of the rules that produced each candidate, but also the extent to which they preserve grammaticality and meaning in the particular context of the input sentence, as well as the degree to which the candidate differs from the input. We experimented with both a Maximum Entropy classifier and an SVR ranker. Experimental results show that incorporating features from an existing paraphrase recognizer in the ranking component improves performance, and that our overall method compares well against a state of the art paraphrase generator, when paraphrasing rules apply to the input sentences. We also propose a new methodology to evaluate the ranking components of generate-and-rank paraphrase generators, which evaluates them across different combinations of weights for grammaticality, meaning preservation, and diversity. The paper is accompanied by a paraphrasing dataset we constructed for evaluations of this kind.

1 Introduction

In recent years, significant effort has been devoted to research on paraphrasing (Androutsopoulos and Malakasiotis, 2010; Madnani and Dorr, 2010). The methods that have been proposed can be roughly classified into three categories: (i) *recognition* methods, i.e., methods that detect whether or not two input sentences or other texts are paraphrases; (ii) *generation* methods, where the aim is to produce paraphrases of a given input sentence; and (iii) *extraction* methods, which aim to extract paraphrasing rules (e.g., "X wrote Y" "$\leftrightarrow$ Y was authored by X") or similar patterns from corpora. Most of the methods that have been proposed belong in the first category, possibly because of the thrust provided by related research on textual entailment recognition (Dagan et al., 2009), where the goal is to decide whether or not the information of a given text is entailed by that of another. Significant progress has also been made in paraphrase extraction, where most recent methods produce large numbers of paraphrasing rules from multilingual parallel corpora (Bannard and Callison-Burch, 2005; Callison-Burch, 2008; Zhao et al., 2008; Zhao et al., 2009a; Zhao et al., 2009b; Kok and Brockett, 2010). In this paper, we are concerned with paraphrase generation, which has received less attention than the other two categories.

There are currently two main approaches to paraphrase generation. The first one treats paraphrase generation as a machine translation problem, with the peculiarity that the target language is the same as the source one. To bypass the lack of large *monolingual* parallel corpora, which are needed to train statistical machine translation (SMT) systems for paraphrasing, monolingual clusters of news articles referring to the same event (Quirk et al., 2004) or other similar monolingual comparable corpora can be used, though sentence alignment methods for parallel corpora may perform poorly on comparable corpora (Nelken and Shieber, 2006); alternatively, large collections of paraphrasing rules obtained via paraphrase extraction from multilingual parallel corpora can be used as monolingual phrase tables in a

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 96–106,
Edinburgh, Scotland, UK, July 27–31, 2011.

phrase-based SMT systems (Zhao et al., 2008; Zhao et al., 2009a); in both cases, paraphrases can then be generated by invoking an SMT system's decoder (Koehn, 2009). A second paraphrase generation approach is to treat existing machine translation engines as black boxes, and translate each input sentence to a pivot language and then back to the original language (Duboue and Chu-Carroll, 2006). An extension of this approach uses multiple translation engines and pivot languages (Zhao et al., 2010).

In this paper, we investigate a different paraphrase generation approach, which does not produce paraphrases by invoking machine translation system(s). We use an existing collection of monolingual paraphrasing rules extracted from multilingual parallel corpora (Zhao et al., 2009b); each rule is accompanied by one or more scores, intended to indicate the rule's *overall* quality without considering particular contexts where the rule may be applied. Instead of using the rules as a monolingual phrase table and invoking an SMT system's decoder, we follow a generate and rank approach, which is increasingly common in several language processing tasks.[1] Given an input sentence, we use the paraphrasing rules to generate a large number of candidate paraphrases. The candidates are then represented as feature vectors, and a ranker (or classifier) selects the best ones; we experimented with a Maximum Entropy classifier and a Support Vector Regression (SVR) ranker.

The vector of each candidate paraphrase includes features indicating the overall quality of the rules that produced the candidate, the extent to which the rules preserve grammaticality and meaning in the particular context of the input sentence, and the degree to which the candidate's surface form differs from that of the input; we call the latter factor *diversity*. The intuition is that a good paraphrase is grammatical, preserves the meaning of the original sentence, while also being as different as possible.

Experimental results show that including in the ranking (or classification) component features from an existing paraphrase recognizer leads to improved results. We also propose a new methodology to evaluate the ranking components of generate-and-rank paraphrase generators, which evaluates them across different combinations of weights for grammaticality, meaning preservation, and diversity. The paper is accompanied by a new publicly available paraphrasing dataset we constructed for evaluations of this kind. Further experiments indicate that when paraphrasing rules apply to the input sentences, our paraphrasing method is competitive to a state of the art paraphrase generator that uses multiple translation engines and pivot languages (Zhao et al., 2010).

We note that paraphrase generation is useful in several language processing tasks. In question answering, for example, paraphrase generators can be used to paraphrase the user's queries (Duboue and Chu-Carroll, 2006; Riezler and Liu, 2010); and in machine translation, paraphrase generation can help improve the translations (Callison-Burch et al., 2006; Marton et al., 2009; Mirkin et al., 2009; Madnani et al., 2007), or it can be used when evaluating machine translation systems (Lepage and Denoual, 2005; Zhou et al., 2006; Kauchak and Barzilay, 2006; Padó et al., 2009).

The remainder of this paper is structured as follows: Section 2 explains how our method generates candidate paraphrases; Section 3 introduces the dataset we constructed, which is also used in subsequent sections; Section 4 discusses how candidate paraphrases are ranked; Section 5 compares our overall method to a state of the art paraphrase generator; and Section 6 concludes.

2 Generating candidate paraphrases

We use the approximately one million English paraphrasing rules of Zhao et al. (2009b). Roughly speaking, the rules were extracted from a parallel English-Chinese corpus, based on the assumption that two English phrases e_1 and e_2 that are often aligned to the same Chinese phrase c are likely to be paraphrases and, hence, they can be treated as a paraphrasing rule $e_1 \leftrightarrow e_2$.[2] Zhao et al.'s method actually operates on slotted English phrases, obtained from parse trees, where slots correspond to part of speech (POS) tags. Hence, rules like the following three may be obtained, where NN_i indicates a noun slot and NNP_i a proper name slot.

[1] See, for example, Collins and Koo (2005).

[2] This pivot-based paraphrase extraction approach was first proposed by Bannard and Callison-Burch (2005). It underlies several other paraphrase extraction methods (Riezler et al., 2007; Callison-Burch, 2008; Kok and Brockett, 2010).

(1) a lot of NN_1 ↔ plenty of NN_1
(2) NNP_1 area ↔ NNP_1 region
(3) NNP_1 wrote NNP_2 ↔ NNP_2 was written by NNP_1

In the basic form of their method, called Model 1, Zhao et al. (2009b) use a log-linear ranker to assign scores to candidate English paraphrase pairs $\langle e_1, e_2 \rangle$; the ranker uses the alignment probabilities $P(c|e_1)$ and $P(e_2|c)$ as features, along with features that assess the quality of the corresponding alignments. In an extension of their method, Model 2, Zhao et al. consider two English phrases e_1 and e_2 as paraphrases, if they are often aligned to two Chinese phrases c_1 and c_2, which are themselves paraphrases according to Model 1 (with English used as the pivot language). Again, a log-linear ranker assigns a score to each $\langle e_1, e_2 \rangle$ pair, now with $P(c_1|e_1)$, $P(c_2|c_1)$, and $P(e_2|c_1)$ as features, along with similar features for alignment quality. In a further extension, Model 3, all the candidate phrase pairs $\langle e_1, e_2 \rangle$ are collectively treated as a monolingual parallel corpus. The phrases of the corpus are aligned, as when aligning a bilingual parallel corpus, and additional features, based on the alignment, are added to the log-linear ranker, which again assigns a score to each $\langle e_1, e_2 \rangle$.

The resulting paraphrasing rules $e_1 \leftrightarrow e_2$ typically contain short phrases (up to four or five words excluding slots) on each side; hence, they can be used to rewrite only parts of longer sentences. Given an input (source) sentence S, we generate candidate paraphrases by applying rules whose left or right hand side matches any part of S. For example, rule (1) matches the source sentence (4); hence, (4) can be rewritten as the candidate paraphrase (5).[3]

(4) S: He had a lot of [$_{NN_1}$admiration] for his job.
(5) C: He had plenty of [$_{NN_1}$admiration] for his job.

Several rules may apply to S; for example, they may rewrite different parts of S, or they may replace the same parts of S by different phrases. We allow all possible combinations of applicable rules to apply to S, excluding combinations that include rules rewriting overlapping parts of S.[4] To avoid generating too many candidates (C), we use only the 20 rules (that apply to S) with the highest scores. Zhao et al. actually associate each rule with three scores. The first one, hereafter called r_1, is the Model 1 score, and the other two, r_2 and r_3, are the forward and backward alignment probabilities of Model 3; see Zhao et al. (2009b) for details. We use the average of the three scores, hereafter r_4, when generating candidates.

Unfortunately, Zhao et al.'s scores reflect the overall quality of each rule, without considering the context of the particular S where the rule is applied. Szpektor et al. (2008) point out that, for example, a rule like "X acquire Y" ↔ "X buy Y" may work well in many contexts, but not in "Children acquire language quickly". Similarly, "X charged Y with" ↔ "X accused Y of" should not be applied to sentences about charging batteries. Szpektor et al. propose, roughly speaking, to associate each rule with a model of the contexts where the rule is applicable, as well as models of the expressions that typically fill its slots, in order to be able to assess the applicability of each rule in specific contexts. The rules that we use do not have associated models of this kind, but we follow Szpektor et al.'s idea of assessing the applicability of each rule in each particular context, when ranking candidates, as discussed below.

3 A dataset of candidate paraphrases

Our generate and rank method relies on existing large collections of paraphrasing rules to generate candidate paraphrases. Our main contribution is in the ranking of the candidates. To be able to evaluate the performance of different rankers in the task we are concerned with, we first constructed an evaluation dataset that contains pairs $\langle S, C \rangle$ of source (input) sentences and candidate paraphrases, and we asked human judges to assess the degree to which the C of each pair was a good paraphrase of S.

We selected randomly 75 source (S) sentences from the AQUAINT corpus, such that at least one of the paraphrasing rules applied to each S.[5] For each S, we generated candidate Cs using Zhao et al.'s rules, as discussed in Section 2. This led to 1,935 $\langle S, C \rangle$ pairs, approx. 26 pairs for each S. The pairs were given to 13 judges other than the authors.[6] Each judge evaluated approx. 148 (different) $\langle S, C \rangle$

[3] We use Stanford's POS tagger, MaxEnt classifier, and dependency parser; see `http://nlp.stanford.edu/`.

[4] A possible extension, which we have not explored, would be to recursively apply the same process to the resulting Cs.

[5] The corpus is available from the LDC (LDC2002T31).

[6] The judges were fluent, but not native English speakers.

Figure 1: Distribution of overall quality scores in the evaluation dataset (1 = totally unacceptable, 4 = perfect).

pairs; each of the 1,935 pairs was evaluated by one judge. The judges were asked to provide grammaticality, meaning preservation, and overall paraphrase quality scores for each $\langle S, C \rangle$ pair, each score on a 1–4 scale (1 for totally unacceptable, 4 for perfect); guidelines and examples were also provided.

Figure 1 shows the distribution of the overall quality scores in the 1,935 $\langle S, C \rangle$ pairs of the evaluation dataset; the distributions of the grammaticality and meaning preservation scores are similar. Notice that although we used only the 20 applicable paraphrasing rules with the highest scores to generate the $\langle S, C \rangle$ pairs, less than half of the candidate paraphrases (C) were considered good, and approximately only 20% perfect. In other words, applying paraphrasing rules (even only those with the 20 best scores) to each input sentence S and randomly picking one of the resulting candidate paraphrases C, without any further filtering (or ranking) of the candidates, would on average produce unacceptable paraphrases more frequently than acceptable ones. Hence, the role of the ranking component is crucial.

We also measured inter-annotator agreement by constructing, in the same way, 100 additional $\langle S, C \rangle$ pairs (other than the 1,935) and asking 3 of the 13 judges to evaluate all of them. We measured the mean absolute error, i.e., the mean absolute difference in the judges' scores (averaged over all pairs of judges) and the mean (over all pairs of judges) K statistic (Carletta, 1996). In the overall scores, K was 0.64, which is in the range often taken to indicate substantial agreement (0.61–0.80).[7] Agreement was higher for grammaticality ($K = 0.81$), and lower ($K = 0.59$) for meaning preservation. Table 1 shows that the mean absolute difference in the annotators' scores was $\frac{1}{5}$ to $\frac{1}{4}$ of a point.

	mean abs. diff.	K-statistic
grammaticality	0.20	0.81
meaning preserv.	0.26	0.59
overall quality	0.22	0.64

Table 1: Inter-annotator agreement when manually evaluating candidate paraphrases.

Several judges commented that they had trouble deciding to what extent the overall quality score should reflect grammaticality or meaning preservation. They also wondered if it was fair to consider as perfect candidate paraphrases that differed in only one or two words from the source sentences, i.e., candidates with low diversity. These comments led us to ignore the judges' overall quality scores in some experiments, and to use a weighted average of grammaticality, meaning preservation, and (automatically measured) diversity instead, with different weight combinations corresponding to different application requirements, as discussed further below.

In the same way, 1,500 more $\langle S, C \rangle$ pairs (other than the 1,935 and the 100, not involving previously seen Ss) were constructed, and they were evaluated by the first author. The 1,500 pairs were used as a training dataset in experiments discussed below. Both the 1,500 training and the 1,935 evaluation (test) pairs are publicly available.[8] We occasionally refer to the training and evaluation datasets as a single dataset, but they are clearly separated.

4 Ranking candidate paraphrases

We now discuss the ranking component of our method, which assesses the candidate paraphrases.

4.1 Features of the ranking component

Each $\langle S, C \rangle$ pair is represented as a feature vector. To allow the ranking component to assess the degree to which a candidate C is grammatical, or at least as grammatical as the source S, we include in the feature vectors the language model scores of S, C, and the difference between the two scores. We use a 3-gram language model trained on approximately

[7] It is also close to 0.67, which is sometimes taken to be a cutoff for substantial agreement in computational linguistics.

[8] See the paper's supplementary material.

6.5 million sentences of the AQUAINT corpus.[9] To allow the ranker to consider the (context-insensitive) quality scores of the rules that generated C from S, we also include as features the highest, lowest, and average r_1, r_2, r_3, and r_4 scores (Section 2) of these rules, 12 features in total.

The features discussed so far are similar to those employed by Zhao et al. (2009a) in the only comparable paraphrase generation method we are aware of that uses paraphrasing rules. That method, hereafter called ZHAO-RUL, uses the language model score of C and scores similar to r_1, r_2, r_3 in a log-linear model.[10] The log-linear model of ZHAO-RUL is used by an SMT-like decoder to identify the transformations (applications of rules) that produce the (hopefully) best paraphrase. By contrast, we first generate a large number of candidates using the paraphrasing rules, and we then rank them. Unfortunately, we did not have access to an implementation of ZHAO-RUL to compare against, but below we compare against another paraphraser proposed by Zhao et al. (2010), hereafter called ZHAO-ENG, which uses multiple machine translation engines and pivot languages, instead of paraphrasing rules, and which Zhao et al. found to outperform ZHAO-RUL.

To further help the ranking component assess the degree to which C preserves the meaning of S, we also optionally include in the vectors of the $\langle S, C\rangle$ pairs the features of an existing paraphrase recognizer (Malakasiotis, 2009) that obtained the best published results (Androutsopoulos and Malakasiotis, 2010) on the widely used MSR paraphrasing corpus.[11] Most of the recognizer's features are computed by using nine similarity measures: Levenshtein, Jaro-Winkler, Manhattan, Euclidean, and n-gram ($n = 3$) distance, cosine similarity, Dice, Jaccard, and matching coefficients, all computed on tokens; consult Malakasiotis (2009) for details. For each $\langle S, C\rangle$ pair, the nine similarity measures are applied to ten different forms $\langle s_1, c_1\rangle, \ldots, \langle s_{10}, c_{10}\rangle$ of $\langle S, C\rangle$, described below, leading to 90 features.

$\langle s_1, c_1\rangle$: The original forms of S and C.

$\langle s_2, c_2\rangle$: S and C with tokens replaced by stems.

$\langle s_3, c_3\rangle$: S and C, with tokens replaced by POS tags.

$\langle s_4, c_4\rangle$: S and C, tokens replaced by soundex codes.[12]

$\langle s_5, c_5\rangle$: S and C, but having removed non-nouns.

$\langle s_6, c_6\rangle$: As previously, but nouns replaced by stems.

$\langle s_7, c_7\rangle$: As previously, nouns replaced by soundex.

$\langle s_8, c_8\rangle$: S and C, but having removed non-verbs.

$\langle s_9, c_9\rangle$: As previously, but verbs replaced by stems.

$\langle s_{10}, c_{10}\rangle$: As previously, verbs replaced by soundex.

When constructing all ten forms $\langle s_i, c_i\rangle$ of $\langle S, C\rangle$, synonyms (in any WordNet synset) are treated as identical words. Additional variants of some of the 90 features compare a sliding window of some of the s_i forms to the corresponding c_i forms (or vice versa), adding 40 more features; see Malakasiotis (2009). Two more Boolean features indicate the existence or absence of negation in S or C, respectively; and another feature computes the ratio of the lengths of S and C, measured in tokens. Finally, three additional features compare the dependency trees of S and C:

$$R_S = \frac{|\text{common dependencies of } S, C|}{|\text{dependencies of } S|}$$

$$R_C = \frac{|\text{common dependencies of } S, C|}{|\text{dependencies of } C|}$$

$$F_{\beta=1} = \frac{2 \cdot R_S \cdot R_C}{R_S + R_C}$$

The recognizer's features are 136 in total.[13] Hence, the full feature set of our paraphraser's ranking component comprises 151 features.

[9] We use SRILM; see `http://www-speech.sri.com/`.

[10] Application-specific features are also included, which can be used, for example, to favor paraphrases that are shorter than the input in sentence compression (Knight and Marcu, 2002; Clarke and Lapata, 2008). Similar features could also be added to application-specific versions of our method.

[11] The MSR corpus contains pairs that are paraphrases or not. It is a benchmark for paraphrase recognizers, not generators. It provides only one paraphrase (true or false) of each source, and few of the true paraphrases can be obtained by the rules we use.

[12] The Soundex algorithm maps English words to alphanumeric codes, so that words with the same pronunciations receive the same codes, despite spelling differences; see `http://en.wikipedia.org/wiki/Soundex`.

[13] Malakasiotis (2009) shows that although there is a lot of redundancy in the recognizer's feature set, the full feature set still leads to better paraphrase recognition results, compared to subsets constructed via feature selection with hill-climbing or beam search. The same paper reports that the recognizer performs almost as well without the last three features, which may not be available in languages with no reliable dependency parsers. Notice, also, that the recognizer does not use paraphrasing rules.

4.2 Learning rate with a MaxEnt classifier

To obtain a first indication of whether or not a ranking component equipped with the features discussed above could learn to distinguish good from bad candidate paraphrases, and to investigate if our training dataset is sufficiently large, we initially experimented with a Maximum Entropy classifier (with the 151 features) as the ranking component. This initial version of the ranking component, called ME-REC, was trained on increasingly larger parts of the training dataset of Section 3, and it was always evaluated on the entire test dataset of that section. For simplicity, we used only the judges' overall quality scores in these experiments, and we treated the problem as one of binary classification; overall quality scores of 1 and 2 where conflated to a negative category, and scores of 3 and 4 to a positive category.

Figure 2 plots the error rate of ME-REC, computed both on the test set and the encountered training subset. The error rate on the training instances a learner has encountered is typically lower than the error rate on the test set (unseen instances); hence, the former error rate can be seen as a lower bound of the latter. ME-REC shows signs of having reached its lower bound when the entire training dataset is used, suggesting that the training dataset is sufficiently large. The baseline (BASE) of Figure 2 uses only a threshold on the average r_4 (Section 2) of the rules that turned S into C. If the average r_4 is higher than the threshold, the $\langle S, C \rangle$ pair is classified in the positive class, otherwise in the negative one. The threshold was tuned by experimenting on a separate tuning dataset. Clearly, ME-REC outperforms the baseline, which uses only the average (context-insensitive) scores of the applied paraphrasing rules.

4.3 Experiments with an SVR ranker

As already noted, when our dataset were constructed the judges felt it was not always clear to what extent the overall quality scores should reflect meaning preservation or grammaticality; and they also wondered if the overall quality scores should have also taken into consideration diversity. To address these concerns, in the experiments described in this section (and the remainder of the paper) we ignored the judges' overall scores, and we used a weighted average of the grammaticality, meaning preservation,

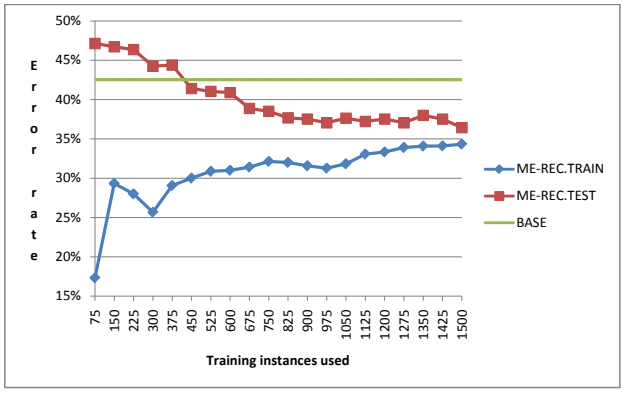


Figure 2: Learning curves of a Maximum Entropy classifier used as the ranking component of our method.

and diversity scores instead; the grammaticality and meaning preservation scores were those provided by the judges, while diversity was automatically computed as the edit distance (Levenshtein, computed on tokens) between S and C. Stated otherwise, the correct score $y(x_i)$ of each training or test instance x_i (i.e., of each feature vector of an $\langle S, C \rangle$ pair) was taken to be a linear combination of the grammaticality score $g(x_i)$, the meaning preservation score $m(x_i)$, and the diversity $d(x_i)$, as in Equation (6), where $\lambda_3 = 1 - \lambda_1 - \lambda_2$.

$$y(x_i) = \lambda_1 \cdot g(x_i) + \lambda_2 \cdot m(x_i) + \lambda_3 \cdot d(x_i) \quad (6)$$

We believe that the λ_i weights should in practice be application-dependent. For example, when paraphrasing user queries to a search engine that turns them into bags of words, diversity and meaning preservation may be more important than grammaticality; by contrast, when paraphrasing the sentences of a generated text to avoid repeating the same expressions, grammaticality is very important. Hence, generic paraphrase generators, like ours, intended to be useful in many different applications, should be evaluated for many different combinations of the λ_i weights. Consequently, in the experiments of this section we trained and evaluated the ranking component of our method (on the training and evaluation part, respectively, of the dataset of Section 3) several times, each time with a different combination of $\lambda_1, \lambda_2, \lambda_3$ values, with the values of each λ_i ranging from 0 to 1 with a step of 0.2.

We employed a Support Vector Regression (SVR) model in the experiments of this section, instead of

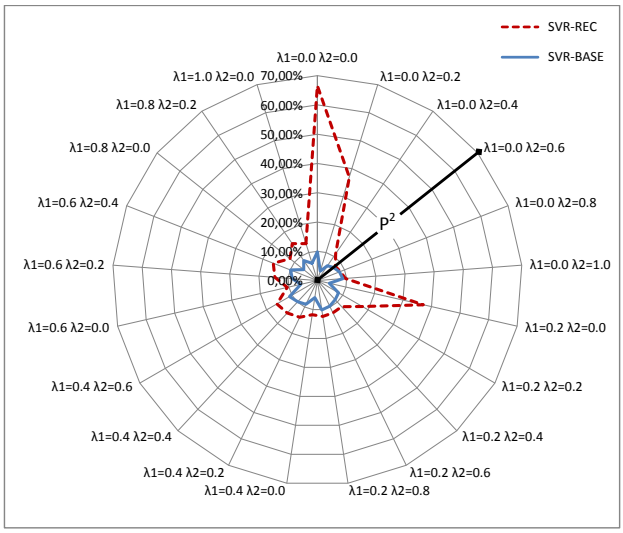


Figure 3: Performance of our method's SVR ranking component with (SVR-REC) and without (SVR-BASE) the additional features of the paraphrase recognizer.

a classifier, given that the $y(x_i)$ scores that we want to predict are real values.[14] An SVR is very similar to a Support Vector Machine (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000; Joachims, 2002), but it is trained on examples of the form $\langle x_i, y(x_i)\rangle$, where $x_i \in \mathbb{R}^n$ and $y(x_i) \in \mathbb{R}$, and it learns a ranking function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that is intended to return $f(x_i)$ values as close as possible to the correct ones $y(x_i)$, given feature vectors x_i. In our case, the correct $y(x_i)$ values were those of Equation (6). We call SVR-REC the SVR ranker with all the 151 features of Section 4.2, and SVR-BASE the SVR ranker without the 136 features of the paraphrase recognizer.

We used the squared correlation coefficient ρ^2 to evaluate SVR-REC against SVR-BASE.[15] The ρ^2 coefficient shows how well the scores returned by the SVR are correlated with the desired scores $y(x_i)$; the higher the ρ^2 the higher the agreement. Figure 3 shows the experimental results. Each line from the diagram's center represents a different experimental setting, i.e., a different combination of λ_1 and λ_2; recall that $\lambda_3 = 1 - \lambda_1 - \lambda_2$. The distance of a method's curve from the center is the method's ρ^2 for that setting. The farther a point is from the center the higher ρ^2 is; hence, methods whose curves are closer to the diagram's outmost perimeter are better. Clearly, SVR-REC (which includes the recognizer's features) outperforms SVR-BASE (which relies only on the language model and the scores of the rules).

The two peaks of SVR-REC's curve are when λ_3 is very high (1 or 0.8), i.e., when $y(x_i)$ is dominated by the diversity score; in these cases, SVR-REC is at a clear advantage, since it includes features for surface string similarity (e.g., Levenshtein distance measured on $\langle s_1, c_1\rangle$), which in effect measure diversity, unlike SVR-BASE. Even when λ_1 is very high (1 or 0.8), i.e., when all or most of the weight is placed on grammaticality, SVR-REC outperforms SVR-BASE, indicating that the extra features in SVR-REC also contribute towards assessing grammaticality; by contrast SVR-BASE relies exclusively on the language model for grammaticality. Unfortunately, when λ_2 is very high (1 or 0.8), i.e., when all or most of the weight is placed on meaning preservation, there is no or very small difference between SVR-REC and SVR-BASE, suggesting that the extra features of the paraphrase recognizer are not as useful to the SVR, when assessing meaning preservation, as we would have hoped. Nevertheless, SVR-REC is overall better than SVR-BASE.

We believe that the dataset of Section 3 and the evaluation methodology summarized by Figure 3 will prove useful to other researchers, who may wish to evaluate other ranking components of generate-and-rank paraphrasing methods against ours, for example with different ranking algorithms or features. Similar datasets of candidate paraphrases can also be created using different collections of paraphrasing rules.[16] The same methodology can then be used to evaluate ranking components on those datasets.

5 Comparison to the state of the art

Having established that SVR-REC is a better configuration of our method's ranker than SVR-BASE, we

[14] Additional experiments confirmed that the SVR performs better than ME-REC as the ranking component. We use the SVR implementation of LIBSVM, available from `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`, with an RBF kernel and default settings. All the features are normalized in $[-1, 1]$, when using SVR or ME-REC.

[15] If n is the number of test pairs, $f(x_i)$ the score returned by the SVR for the i-th pair, and $y(x_i)$ the correct score, then ρ^2 is:

$$\frac{(n\sum_{i=1}^{n} f(x_i)y_i - \sum_{i=1}^{n} f(x_i)\sum_{i=1}^{n} y(x_i))^2}{(n\sum_{i=1}^{n} f(x_i)^2 - (\sum_{i=1}^{n} f(x_i))^2)(n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y(x_i))^2)}$$

[16] See Androutsopoulos and Malakasiotis (2010) for pointers.

proceed to investigate how well our overall generate-and-rank method (with SVR-REC) compares against a state of the art paraphrase generator.

As already mentioned, Zhao et al. (2010) recently presented a method (we call it ZHAO-ENG) that outperforms their previous method (Zhao et al., 2009a), which used paraphrasing rules and an SMT-like decoder (we call that previous method ZHAO-RUL). Given an input sentence S, ZHAO-ENG produces candidate paraphrases by translating S to 6 pivot languages via 3 different commercial machine translation engines (treated as black boxes) and then back to the original language, again via 3 machine translation engines (54 combinations). Roughly speaking, ZHAO-ENG then ranks the candidate paraphrases by their average distance from all the other candidates, selecting the candidate(s) with the smallest distance; distance is measured as BLEU score (Papineni et al., 2002).[17] Hence, ZHAO-ENG is also, in effect, a generate-and-rank paraphraser, but the candidates are generated by invoking multiple machine translation engines instead of applying paraphrasing rules, and they are ranked by the average distance measure rather than using an SVR.

An obvious practical advantage of ZHAO-ENG is that it exploits the vast resources of existing commercial machine translation engines when generating candidate paraphrases, which allows it to always obtain large numbers of candidate paraphrases. By contrast, the collection of paraphrasing rules that we currently use does not manage to produce any candidate paraphrases in 40% of the sentences of the New York Times part of AQUAINT, because no rule applies. Hence, in terms of ability to always paraphrase the input, ZHAO-ENG is clearly better, though it should be possible to improve our methods's performance in that respect by using larger collections of paraphrasing rules.[18] A further interesting question, however, is how good the paraphrases of the two methods are, when both methods manage to paraphrase the input, i.e., when at least one paraphrasing rule applies to S. This scenario can be seen as an emulation of the case where the collection of paraphrasing rules is sufficiently large to guarantee that at least one rule applies to any source sentence.

To answer the latter question, we re-implemented ZHAO-ENG, with the same machine translation engines and languages used by Zhao et al. (2010). We also trained our paraphraser (with SVR-REC) on the training part of the dataset of Section 3. We then selected 300 random source sentences S from AQUAINT that matched at least one of the paraphrasing rules, excluding sentences that had been used before. Then, for each one of the 300 S sentences, we kept the single best candidate paraphrase C_1 and C_2, respectively, returned by our paraphraser and ZHAO-ENG. The resulting $\langle S, C_1 \rangle$ and $\langle S, C_2 \rangle$ pairs were given to 10 human judges. This time the judges assigned only grammaticality and meaning preservation scores (on a 1–4 scale); diversity was again computed as edit distance. Each pair was evaluated by one judge, who was given an equal number of pairs from the two methods, without knowing which method each pair came from. The same judge never rated two pairs with the same S. Since we had no way to make ZHAO-ENG sensitive to $\lambda_1, \lambda_2, \lambda_3$, we trained SVR-REC with $\lambda_1 = \lambda_2 = 1/3$, as the most neutral combination of weights.

Table 2 lists the average grammaticality, meaning preservation, and diversity scores of the two methods. All scores were normalized in $[0, 1]$, but the reader should keep in mind that diversity was computed as edit distance, whereas the other two scores were provided by human judges on a 1–4 scale. The grammaticality score of our method was better than ZHAO-ENG's, and the difference was statistically significant.[19] In meaning preservation, ZHAO-ENG was slightly better, but the difference was not statistically significant. The difference in diversity was larger and statistically significant, with the diversity scores indicating that it takes approximately twice as many edit operations (insert, delete, replace) to turn each source sentence to ZHAO-ENG's paraphrase, compared to the paraphrase of our method.

We note that our method can be tuned, by adjusting the λ_i weights, to produce paraphrases with

[17] We use the version of ZHAO-ENG that Zhao et al. (2010) call "selection-based", since they reported it performs overall better than an alternative decoding-based version.

[18] Recall that the paraphrasing rules we use were extracted from an English-Chinese parallel corpus. Additional rules could be extracted from other parallel corpora, like Europarl (`http://www.statmt.org/europarl/`).

[19] We used Analysis of Variance (ANOVA) (Fisher, 1925), followed by post-hoc Tukey tests to check whether the scores of the two methods differ significantly ($p < 0.05$).

score (%)	our method	ZHAO-ENG
grammaticality	**90.89**	85.33
meaning preserv.	76.67	78.56
diversity	6.50	**14.58**

Table 2: Evaluation of our paraphrasing method (with SVR-REC) against ZHAO-ENG, using human judges. Results in bold indicate statistically significant differences.

higher grammaticality, meaning preservation, or diversity scores; for example, we could increase λ_3 and decrease λ_1 to obtain higher diversity at the cost of lower grammaticality in the results of Table 2.[20] It is unclear how ZHAO-ENG could be tuned that way.

Overall, our method seems to perform well against ZHAO-ENG, despite the vastly larger resources of ZHAO-ENG, provided of course that we limit ourselves to source sentences to which paraphrasing rules apply. It would be interesting to investigate in future work if our method's coverage (sentences it can paraphrase) can increase to ZHAO-ENG's level by using larger collections of paraphrasing rules. It would also be interesting to combine the two methods, perhaps by using SVR-REC (without features for the quality scores of the rules) to rank candidate paraphrases generated by ZHAO-ENG.

6 Conclusions and future work

We presented a generate-and-rank method to paraphrase sentences. The method first produces candidate paraphrases by applying existing paraphrasing rules extracted from parallel corpora, and it then ranks (or classifies) the candidates to keep the best ones. The ranking component considers not only the context-insensitive quality scores of the paraphrasing rules that produced each candidate, but also features intended to measure the extent to which the rule applications preserve grammaticality and meaning in the particular context of the input sentence, as well as the degree to which the resulting candidate differs from the input sentence (diversity).

Initial experiments with a Maximum Entropy classifier confirmed that the features we use can help a ranking component select better candidate paraphrases than a baseline ranker that considers only the average context-insensitive quality scores of the applied rules. Further experiments with an SVR ranker indicated that our full feature set, which includes features from an existing paraphrase recognizer, leads to improved performance, compared to a smaller feature set that includes only the context-insensitive scores of the rules and language modeling scores. We also propose a new methodology to evaluate the ranking components of generate-and-rank paraphrase generators, which evaluates them across different combinations of weights for grammaticality, meaning preservation, and diversity. The paper is accompanied by a paraphrasing dataset we constructed for evaluations of this kind.

Finally, we evaluated our overall method against a state of the art sentence paraphraser, which generates candidates by using several commercial machine translation systems and pivot languages. Overall, our method performed well, despite the vast resources of the machine translation systems employed by the system we compared against. Our method performed better in terms of grammaticality, equally well in meaning preservation, and worse in diversity, but it could be tuned to obtain higher diversity at the cost of lower grammaticality, whereas it is unclear how the system we compare against could be tuned this way. On the other hand, an advantage of the paraphraser we compared against is that it always produces paraphrases; by contast, our system does not produce paraphrases when no paraphrasing rule applies to the source sentence. Larger collections of paraphrasing rules would be needed to improve our method in that respect.

Apart from obtaining and experimenting with larger collections of paraphrasing rules, it would be interesting to evaluate our method in vivo, for example by embedding it in question answering systems (to paraphrase the questions), in information extraction systems (to paraphrase extraction templates), or in natural language generators (to paraphrase template-like sentence plans). We also plan to investigate the possibility of embedding our SVR ranker in the sentence paraphraser we compared against, i.e., to rank candidates produced by using several machine translation systems and pivot languages, as in ZHAO-ENG.

[20] Additional application-specific experiments confirm that this tuning is possible (Malakasiotis, 2011).

Acknowledgments

This work was partly carried out during INDIGO, an FP6 IST project funded by the European Union, with additional funding from the Greek General Secretariat of Research and Technology.[21]

References

I. Androutsopoulos and P. Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38:135–187.

C. Bannard and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proc. of the 43rd* ACL, pages 597–604, Ann Arbor, MI.

C. Callison-Burch, P. Koehn, and M. Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proc. of* HLT-NAACL, pages 17–24, New York, NY.

C. Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proc. of* EMNLP, pages 196–205, Honolulu, HI, October.

J. Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22:249–254.

J. Clarke and M. Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 1(31):399–429.

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

N. Cristianini and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

I. Dagan, B. Dolan, B. Magnini, and D. Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Lang. Engineering*, 15(4):i–xvii. Editorial of the special issue on Textual Entailment.

P. A. Duboue and J. Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proc. of* HLT-NAACL, pages 33–36, New York, NY.

Ronald A. Fisher. 1925. *Statistical Methods for Research Workers*. Oliver and Boyd.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines: Methods, Theory, Algorithms*. Kluwer.

D. Kauchak and R. Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proc. of* HLT-NAACL, pages 455–462, New York, NY.

[21]Consult `http://www.ics.forth.gr/indigo/`.

K. Knight and D. Marcu. 2002. Summarization beyond sentence extraction: A probalistic approach to sentence compression. *Artif. Intelligence*, 139(1):91–107.

P. Koehn. 2009. *Statistical Machine Translation*. Cambridge University Press.

S. Kok and C. Brockett. 2010. Hitting the right paraphrases in good time. In *Proc. of* HLT-NAACL, pages 145–153, Los Angeles, CA.

Y. Lepage and E. Denoual. 2005. Automatic generation of paraphrases to be used as translation references in objective evaluation measures of machine translation. In *Proc. of the 3rd Int. Workshop on Paraphrasing*, pages 57–64, Jesu Island, Korea.

N. Madnani and B.J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

N. Madnani, F. Ayan, P. Resnik, and B. J. Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proc. of 2nd Workshop on Statistical Machine Translation*, pages 120–127, Prague, Czech Republic.

P. Malakasiotis. 2009. Paraphrase recognition using machine learning to combine similarity measures. In *Proc. of the Student Research Workshop of* ACL-AFNLP, Singapore.

P. Malakasiotis. 2011. *Paraphrase and Textual Entailment Recognition and Generation*. Ph.D. thesis, Department of Informatics, Athens University of Economics and Business, Greece.

Y. Marton, C. Callison-Burch, and P. Resnik. 2009. Improved statistical machine translation using monolingually-derived paraphrases. In *Proc. of* EMNLP, pages 381–390, Singapore.

S. Mirkin, L. Specia, N. Cancedda, I. Dagan, M. Dymetman, and I. Szpektor. 2009. Source-language entailment modeling for translating unknown terms. In *Proc. of* ACL-AFNLP, pages 791–799, Singapore.

R. Nelken and S. M. Shieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. In *Proc. of the 11th* EACL, pages 161–168, Trento, Italy.

S. Padó, M. Galley, D. Jurafsky, and C. D. Manning. 2009. Robust machine translation evaluation with entailment features. In *Proc. of* ACL-AFNLP, pages 297–305, Singapore.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of the 40th* ACL, pages 311–318, Philadelphia, PA.

C. Quirk, C. Brockett, and W. B. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proc. of the Conf. on* EMNLP, pages 142–149, Barcelona, Spain.

S. Riezler and Y. Liu. 2010. Query rewriting using monolingual statistical machine translation. *Computational Linguistics*, 36(3):569–582.

S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proc. of the 45th* ACL, pages 464–471, Prague, Czech Republic.

I. Szpektor, I. Dagan, R. Bar-Haim, and J. Goldberger. 2008. Contextual preferences. In *Proc. of* ACL-HLT, pages 683–691, Columbus, OH.

V. Vapnik. 1998. *Statistical learning theory*. John Wiley.

S. Zhao, H. Wang, T. Liu, and S. Li. 2008. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *Proc. of* ACL-HLT, pages 780–788, Columbus, OH.

S. Zhao, X. Lan, T. Liu, and S. Li. 2009a. Application-driven statistical paraphrase generation. In *Proc. of* ACL-AFNLP, pages 834–842, Singapore.

S. Zhao, H. Wang, T. Liu, and Li. S. 2009b. Extracting paraphrase patterns from bilingual parallel corpora. *Natural Language Engineering*, 15(4):503–526.

S. Zhao, H. Wang, X. Lan, and T. Liu. 2010. Leveraging multiple MT engines for paraphrase generation. In *Proceedings of the 23rd* COLING, pages 1326–1334, Beijing, China.

L. Zhou, C.-Y. Lin, and Eduard Hovy. 2006. Re-evaluating machine translation results with paraphrase support. In *Proc. of the Conf. on* EMNLP, pages 77–84.

Correcting Semantic Collocation Errors with L1-induced Paraphrases

Daniel Dahlmeier[1] and **Hwee Tou Ng**[1,2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, National University of Singapore
{danielhe,nght}@comp.nus.edu.sg

Abstract

We present a novel approach for automatic collocation error correction in learner English which is based on paraphrases extracted from parallel corpora. Our key assumption is that collocation errors are often caused by semantic similarity in the first language (L1-language) of the writer. An analysis of a large corpus of annotated learner English confirms this assumption. We evaluate our approach on real-world learner data and show that L1-induced paraphrases outperform traditional approaches based on edit distance, homophones, and WordNet synonyms.

1 Introduction

Grammatical error correction (GEC) is emerging as a commercially attractive application of natural language processing (NLP) for the booming market of English as foreign or second language (EFL/ESL[1]).

The de facto standard approach to GEC is to build a statistical model that can choose the most likely correction from a *confusion set* of possible correction choices. The way the confusion set is defined depends on the type of error. Work in context-sensitive spelling error correction (Golding and Roth, 1999) has traditionally focused on confusion sets with similar spelling (e.g., {*dessert*, *desert*}) or similar pronunciation (e.g., {*there*, *their*}). In other words, the words in a confusion set are deemed confusable because of orthographic or phonetic similarity. Other work in GEC has defined the confusion sets based on syntactic similarity, for example all English articles or the most frequent English prepositions form a confusion set (see for example (Tetreault et al., 2010; Rozovskaya and Roth, 2010; Gamon, 2010; Dahlmeier and Ng, 2011) among others).

In contrast, we investigate in this paper a class of grammatical errors where the source of confusion is the similar *semantics* of the words, rather than orthography, phonetics, or syntax. In particular, we focus on *collocation errors* in EFL writing. The term *collocation* (Firth, 1957) describes a sequence of words that is conventionally used together in a particular way by native speakers and appears more often together than one would expect by chance. The correct use of collocations is a major difficulty for EFL students (Farghal and Obiedat, 1995).

In this work, we present a novel approach for automatic correction of collocation errors in EFL writing. Our key observation is that words are potentially confusable for an EFL student if they have similar translations in the writer's first language (L1-language), or in other words if they have the same semantics in the L1-language of the writer. The Chinese word 看 (*kàn*), for example, has over a dozen translations in English, including the words *see*, *look*, *read*, and *watch*. A Chinese speaker who still "thinks" in Chinese has to choose from all these possible translations when he wants to express a sentence like *I like to **watch** movies* and might instead produce a sentence like **I like to **look** movies*. Although the meanings of *watch* and *look* are similar, the former is clearly the more fluent choice in this context. While these types of *L1-transfer er-*

[1]For simplicity, we will collectively refer to both terms as *English as a foreign language (EFL)*

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 107–117,
Edinburgh, Scotland, UK, July 27–31, 2011.

rors have been known in the EFL teaching literature (Swan and Smith, 2001; Meng, 2008), research in GEC has mostly ignored this fact.

We first analyze collocation errors in the NUS Corpus of Learner English (NUCLE), a fully annotated one-million-word corpus of learner English which we will make available to the community for research purposes (see Section 3 for details about the corpus). Our analysis confirms that many collocation errors can be traced to similar translations in the writer's L1-language. Based on this result, we propose a novel approach for automatic collocation error correction. The key component in our approach generates *L1-induced paraphrases* which we automatically extract from an L1-English parallel corpus. Our proposed approach outperforms traditional approaches based on edit distance, homophones, and WordNet synonyms on a test set of real-world learner data in an automatic and a human evaluation. Finally, we present a detailed analysis of unsolved instances in our data set to highlight directions for future work.

Our work adds to a growing body of research that leverages parallel corpora for semantic NLP tasks, for example in word sense disambiguation (Ng et al., 2003; Chan and Ng, 2005; Ng and Chan, 2007; Zhong and Ng, 2009), paraphrasing (Bannard and Callison-Burch, 2005; Liu et al., 2010a), and machine translation evaluation (Snover et al., 2009; Liu et al., 2010b).

The remainder of this paper is organized as follows. The next section reviews related work. Section 3 presents our analysis of collocation errors. Section 4 describes our approach for automatic collocation error correction. The experimental setup and the results are described in Sections 5 and 6, respectively. Section 7 provides further analysis. Section 8 concludes the paper.

2 Related Work

In this section, we give an overview of related work on collocation error correction. We also highlight differences between collocation error correction and related NLP tasks like context-sensitive spelling error correction, synonym extraction, lexical substitution, and paraphrasing.

Most work in collocation error correction has relied on dictionaries or manually created databases to generate collocation candidates (Shei and Pain, 2000; Wible et al., 2003; Futagi et al., 2008). Other work has focused on finding candidates that collocate with similar words, e.g., verbs that appear with the same noun objects form a confusion set (Liu et al., 2009; Wu et al., 2010). The work most similar to ours is probably the one presented by Chang *et al.* (2008), as they also use translation information to generate collocation candidates. However, they do not use automatically derived paraphrases from parallel corpora but bilingual dictionaries. Dictionaries usually have lower coverage, do not contain longer phrases or inflected forms, and do not provide any translation probability estimates. Also, their work focuses solely on verb-noun collocations, while we target collocations of arbitrary syntactic type.

Context-sensitive spelling error correction is the task of correcting spelling mistakes that result in another valid word, see for example (Golding and Roth, 1999). It has traditionally focused on a small number of pre-defined confusion sets, like homophones or frequent spelling errors. Even when the confusion sets are formed automatically, the similarity of words in a confusion set has been based on edit distance or phonetic similarity (Carlson et al., 2001). In contrast, we focus on words that are confusable due to their similar semantics instead of similar spelling or pronunciation. Also, we do not assume that the set of confusion sets is already given to us. Instead, we automatically extract confusable candidates from a parallel corpus.

Synonym extraction (Wu and Zhou, 2003), lexical substitution (McCarthy and Navigli, 2007) and paraphrasing (Madnani and Dorr, 2010) are related to collocation correction in the sense that they try to find semantically equivalent words or phrases. However, there is a subtle but important difference between these tasks and collocation correction. In the former, the main criterion is whether the original phrase and the synonym/paraphrase candidate are substitutable, i.e., both form a grammatical sentence when substituted for each other in a particular context. In contrast, in collocation correction, we are primarily interested in finding candidates which are *not substitutable* in their English context but *appear to be substitutable* in the L1-language of the writer, i.e., one forms a grammatical English sentence but the other does not.

Sentences	52,149
Words	1,149,100
Distinct words	27,593
Avg. sentence length (words)	22.04
Collocation errors	2,747
Avg. collocation error length (words)	1.17
Avg. correction length (words)	1.13

Table 1: Statistics of the NUS Corpus of Learner English (NUCLE)

3 Analysis of EFL collocation errors

While the fact that collocation errors can be caused by L1-transfer has been ascertained by EFL researchers (Meng, 2008), we need to quantify how frequent collocation errors can be traced to these types of transfer errors in order to estimate how many errors in EFL writing we can potentially hope to correct with information about the writer's L1-language.

We base our analysis on the NUS Corpus of Learner English (NUCLE). The corpus consists of about 1,400 essays written by EFL university students on a wide range of topics, like environmental pollution or healthcare. Most of the students are native Chinese speakers. The corpus contains over one million words which are completely annotated with error tags and corrections. All annotations have been performed by professional English instructors. The statistics of the corpus are summarized in Table 1. The annotation is stored in a stand-off fashion. Each error tag consists of the start and end offset of the annotation, the type of the error, and the appropriate gold correction as deemed by the annotator. The annotators were asked to provide a correction that would result in a grammatical sentence if the selected word or phrase would be replaced by the correction.

In this work, we focus on errors which have been marked with the error tag *wrong collocation/idiom/preposition*. As preposition errors are not the focus of this work, we automatically filter out all instances which represent simple substitutions of prepositions, using a fixed list of frequent English prepositions. In a similar way, we filter out a small number of article errors which were marked as collocation errors. Finally, we filter out instances where the annotated phrase or the suggested correction is longer than 3 words, as we observe that they contain highly context-specific corrections and are unlikely to generalize well (e.g., *"for the simple reasons that these can help them"* $\rightarrow$ *"simply to"*).

After filtering, we end up with 2,747 collocation errors and their respective corrections, which account for about 6% of all errors in NUCLE. This makes collocation errors the 7th largest class of errors in the corpus after article errors, redundancies, prepositions, noun number, verb tense, and mechanics. Not counting duplicates, there are 2,412 distinct collocation errors and corrections. Although there are other error types which are more frequent, collocation errors represent a particular challenge as the possible corrections are not restricted to a closed set of choices and they are directly related to *semantics* rather than syntax. We analyzed the collocation errors and found that they can be attributed to the following sources of confusion:

Spelling: We suspect that an error is caused by similar orthography if the edit distance between the erroneous phrase and its correction is less than a certain threshold.

Homophones: We suspect that an error is caused by similar pronunciation if the erroneous word and its correction have the same pronunciation. We use the CuVPlus English dictionary (Mitton, 1992) to map words to their phonetic representations.

Synonyms: We suspect that an error is caused by synonymy if the erroneous word and its correction are synonyms in WordNet (Fellbaum, 1998). We use WordNet 3.0.

L1-transfer: We suspect that an error is caused by L1-transfer if the erroneous phrase and its correction share a common translation in a Chinese-English phrase table. The details of the phrase table construction are described in Section 4. We note that although we focus on Chinese-English translation, our method is applicable to any language pair where parallel corpora are available.

As CuVPlus and WordNet are defined for individual words, we extend the matching process to phrases in the following way: two phrases A and B are deemed homophones/synonyms if they have the same length and the i-th word in phrase A is a homophone/synonym of the corresponding i-th word in phrase B.

Spelling	... it received *critics (criticism)* as much as complaints ...
	... budget for the aged to *improvise (improve)* other areas.
Homophones	... diverse spending can *aide (aid)* our country.
	... *insure (ensure)* the safety of civilians ...
Synonyms	... rapid *increment (increase)* of the seniors ...
	... energy that we can *apply (use)* in the future ...
L1-transfer	... and *give (provide,* 给予 *)* reasonable fares to the public ...
	... and *concerns (attention,* 关注 *)* that the nation put on technology and engineering ...

Table 3: Examples of collocation errors with different sources of confusion. The correction is shown in parenthesis. For L1-transfer, we also show the shared Chinese translation. The L1-transfer examples shown here do not belong to any of the other categories.

Suspected Error Source	Tokens	Types
Spelling	154	131
Homophones	2	2
Synonyms	74	60
L1-transfer	1016	782
L1-transfer w/o spelling	954	727
L1-transfer w/o homophones	1015	781
L1-transfer w/o synonyms	958	737
L1-transfer w/o spelling, homophones, synonyms	906	692

Table 2: Analysis of collocation errors. The threshold for spelling errors is one for phrases of up to six characters and two for the remaining phrases.

The results of the analysis are shown in Table 2. Tokens refer to running erroneous phrase-correction pairs including duplicates, and types refer to distinct erroneous phrase-correction pairs. As a collocation error can be part of more than one category, the rows in the table do not sum up to the total number of errors. The number of errors that can be traced to L1-transfer greatly outnumbers all other categories. The table also shows the number of collocation errors that can be traced to L1-transfer but not the other sources. 906 collocation errors with 692 distinct collocation error types can be attributed only to L1-transfer but not to spelling, homophones, or synonyms. Table 3 shows some examples of collocation errors for each category from our corpus. We note that there are also collocation error types that cannot be traced to any of the above sources. We will return to these errors in Section 7.

4 Correcting Collocation Errors

In this section, we propose a novel approach for correcting collocation errors in EFL writing.

4.1 L1-induced Paraphrases

We use the popular technique of paraphrasing with parallel corpora (Bannard and Callison-Burch, 2005) to automatically find collocation candidates from a sentence-aligned L1-English parallel corpus. As most of the essays in our corpus are written by native Chinese speakers, we use the FBIS Chinese-English corpus, which consists of about 230,000 Chinese sentences (8.5 million words) from news articles, each with a single English translation. We tokenize and lowercase the English half of the corpus in the standard way. We segment the Chinese half of the corpus using the maximum entropy segmenter from (Ng and Low, 2004; Low et al., 2005). Subsequently, we automatically align the texts at the word level using the Berkeley aligner (Liang et al., 2006; Haghighi et al., 2009). We extract English-L1 and L1-English phrases of up to three words from the aligned texts using the widely used phrase extraction heuristic in (Koehn et al., 2003). The paraphrase probability of an English phrase e_1 given an English phrase e_2 is defined as

$$p(e_1|e_2) = \sum_f p(e_1|f)p(f|e_2) \qquad (1)$$

where f denotes a foreign phrase in the L1 language. The phrase translation probabilities $p(e_1|f)$ and $p(f|e_2)$ are estimated by maximum likelihood estimation and smoothed using Good-Turing smoothing (Foster et al., 2006). Finally, we only keep para-

phrases with a probability above a certain threshold (set to 0.001 in our work).

4.2 Collocation Correction with Phrase-based SMT

We implement our approach in the framework of phrase-based statistical machine translation (SMT) (Koehn et al., 2003). Phrase-based SMT tries to find the highest scoring translation $\mathbf{e}$ given an input sentence $\mathbf{f}$. The *decoding* process of finding the highest scoring translation is guided by a log-linear model which scores translation candidates using a set of feature functions $h_i,\ i = 1, \ldots, n$

$$score(\mathbf{e}|\mathbf{f}) = \exp\left(\sum_{i=1}^{n} \lambda_i h_i(\mathbf{e}, \mathbf{f})\right) . \quad (2)$$

Typical features include a phrase translation probability $p(e|f)$, an inverse phrase translation probability $p(f|e)$, a language model score $p(\mathbf{e})$, and a constant phrase penalty. The optimization of the feature weights $\lambda_i,\ i = 1, \ldots, n$ can be done using minimum error rate training (MERT) (Och, 2003) on a development set of input sentences and their reference translations.

Because of the great flexibility of the log-linear model, researchers have used the framework for other tasks outside SMT, including grammatical error correction (Brockett et al., 2006). We adopt a similar approach in this work. We modify the phrase table of the popular phrase-based SMT decoder MOSES (Koehn et al., 2007) to include collocation corrections with features derived from spelling, homophones, synonyms, and L1-induced paraphrases.

- **Spelling:** For each English word, the phrase table contains entries consisting of the word itself and each word that is within a certain edit distance from the original word. Each entry has a constant feature of 1.0.

- **Homophones:** For each English word, the phrase table contains entries consisting of the word itself and each of the word's homophones. We determine homophones using the CuVPlus dictionary. Each entry has a constant feature of 1.0.

- **Synonyms:** For each English word, the phrase table contains entries consisting of the word itself and each of its synonyms in WordNet. If a word has more than one sense, we consider all its senses. Each entry has a constant feature of 1.0.

- **L1-paraphrases:** For each English phrase, the phrase table contains entries consisting of the phrase and each of its L1-derived paraphrases as described in Section 4.1. Each entry has two real-valued features: a paraphrase probability according to Equation 1 and an inverse paraphrase probability.

- **Baseline** We combine the phrase tables built for spelling, homophones, and synonyms. The combined phrase table contains three binary features for spelling, homophones, and synonyms, respectively.

- **All** We combine the phrase tables from spelling, homophones, synonyms, and L1-paraphrases. The combined phrase table contains five features: three binary features for spelling, homophones, and synonyms, and two real-valued features for the L1-paraphrase probability and inverse L1-paraphrase probability.

Additionally, each phrase table contains the standard constant phrase penalty feature. The first four tables only contain collocation candidates for individual words. We leave it to the decoder to construct corrections for longer phrases during the decoding process if necessary.

5 Experiments

In this section, we empirically evaluate our approach on real collocation errors in learner English.

5.1 Data Set

We randomly sample a development set of 770 sentences and a test set of 856 sentences from our corpus. Each sentence contains exactly one collocation error. The sampling is performed in a way that sentences from the same document cannot end up in both the development and the test set. In order to

keep conditions as realistic as possible, we make no attempt to filter the test set in any way.

We build phrase tables as described in Section 4.2. For the purpose of the experiments reported in this paper, we only need to generate phrase table entries for words and phrases which actually appear in the development or test set.

5.2 Evaluation Metrics

We conduct an automatic and a human evaluation. Our main evaluation metric is *mean reciprocal rank (MRR)* which is the arithmetic mean of the inverse ranks of the first correct answer returned by the system

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank(i)} \quad (3)$$

where N is the size of the test set. If the system did not return a correct answer for a test instance, we set $\frac{1}{rank(i)}$ to zero.

In the human evaluation, we additionally report precision at rank k, $k = 1, 2, 3$, which we calculate as follows:

$$\text{P@k} = \frac{\sum_{a \in A} score(a)}{|A|} \quad (4)$$

where A is the set of returned answers of rank k or less and $score(\cdot)$ is a real-valued scoring function between zero and one.

5.3 Collocation Error Experiments

Automatic correction of collocation errors can conceptually be divided into two steps: *i) identification* of wrong collocations in the input, and *ii) correction* of the identified collocations. In this work, we focus on the second step and assume that the erroneous collocation has already been identified. While this might seem like a simplification, it has been the common evaluation setup in collocation error correction (see for example (Wu et al., 2010)). It also has a practical application where the user first selects a word or phrase and the system displays possible corrections.

In our experiments, we use the start and end offset of the collocation error provided by the human annotator to identify the location of the collocation error. We fix the translation of the rest of the sentence to its identity. We remove phrase table entries where the phrase and the candidate correction are identical, thus practically forcing the system to change the identified phrase. We set the distortion limit of the decoder to zero to achieve monotone decoding. We previously observed that word order errors are virtually absent in our collocation errors. For the language model, we use a 5-gram language model trained on the English Gigaword corpus with modified Kneser-Ney smoothing. All experiments use the same language model to allow a fair comparison.

We perform MERT training with the popular BLEU metric (Papineni et al., 2002) on the development set of erroneous sentences and their corrections. As the search space is restricted to changing a single phrase per sentence, training converges relatively quickly after two or three iterations. After convergence, the model can be used to automatically correct new collocation errors.

6 Results

We evaluate the performance of the proposed method on our test set of 856 sentences, each with one collocation error. We conduct both an automatic and a human evaluation. In the automatic evaluation, the system's performance is measured by computing the rank of the gold answer provided by the human annotator in the n-best list of the system. We limit the size of the n-best list to the top 100 outputs. If the gold answer is not found in the top 100 outputs, the rank is considered to be infinity, or in other words, the inverse of the rank is zero. We also report the number of test instances for which the gold answer was ranked among the top k answers, $k = 1, 2, 3, 10, 100$. The results of the automatic evaluation are shown in Table 4

For collocation errors, there are usually more than one possible correct answer. Therefore, automatic evaluation underestimates the actual performance of the system by only considering the single gold answer as correct and all other answers as wrong. As such, we carried out a human evaluation for the systems BASELINE and ALL. We recruited two English speakers to judge a subset of 500 test sentences. For each sentence, a judge was shown the original sentence and the 3-best candidates of each of the two systems. We restricted human evaluation to the 3-best candidates, as we believe that answers at a rank

Model	Rank = 1	Rank ≤ 2	Rank ≤ 3	Rank ≤ 10	Rank ≤ 100	MRR
Spelling	35	41	42	44	44	4.51
Homophones	1	1	1	1	1	0.11
Synonyms	32	47	52	60	61	4.98
Baseline	49	68	80	93	96	7.61
L1-paraphrases	93	133	154	**216**	**243**	15.43
All	**112**	**150**	**166**	**216**	241	**17.21**

Table 4: Results of automatic evaluation. Columns two to six show the number of gold answers that are ranked within the top k answers. The last column shows the mean reciprocal rank in percentage. Bigger values are better.

P(A)	0.8076
Kappa	0.6152

Table 5: Inter-annotator agreement. $P(E) = 0.5$.

larger than three will not be very useful in a practical application. The candidates are displayed together in alphabetical order without any information about their rank or which system produced them or the gold answer by the annotator. The difference between the candidates and the original sentence is highlighted. The judges were asked to make a binary judgment for each of the candidates on whether the proposed candidate is a valid correction of the original or not. We represent valid corrections with a score of 1.0 and invalid corrections with a score of 0.0. Inter-annotator agreement is reported in Table 5. The chance of agreement $P(A)$ is the percentage of times that the annotators agree, and $P(E)$ is the expected agreement by chance, which is 0.5 in our case. The Kappa coefficient is defined as

$$\text{Kappa} = \frac{\text{P(A)} - \text{P(E)}}{1 - \text{P(E)}}$$

We obtain a Kappa coefficient of 0.6152. A Kappa coefficient between 0.6 and 0.8 is considered as showing *substantial* agreement according to Landis and Koch (1977). To compute precision at rank k, we average the judgments. Thus, a system can receive a score of 0.0 (both judgments negative), 0.5 (judges disagree), or 1.0 (both judgments positive) for each returned answer. To compute MRR, we cannot simply average the judgments as MRR requires binary judgments on whether an item is correct or not. Instead, we report MRR on the union and the intersection of the judgments. In the first case, the rank of the first correct item is the minimum rank of any item judged correct by *either* judge. In the second case, the rank of the first correct item is the minimum rank of any item judged correct by *both* judges. The results for the human evaluation are shown in Table 6. Our best system ALL outperforms the BASELINE approach on all measures. It receives a precision at rank 1 of 38.20% and a MRR of 33.16% (intersection) and 57.26% (union). Table 7 shows some examples from our test set.

Unfortunately, comparison of our results with previous work is complicated by the fact that there currently exists no standard data set for collocation error correction. We will make our corpus available for research purposes in the hope that it will allow researchers to more directly compare their results in future.

7 Analysis

In this section, we analyze and categorize those test instances for which the ALL system could not produce an acceptable correction in the top 3 candidates. We manually analyze 100 test sentences for which neither judge had deemed any candidate answer to be a valid correction. Based on our findings, we categorize the 100 sentences into eight categories which are shown below. Table 8 shows examples from each category.

Out-of-vocabulary (21/100) The most frequent reason why the system does not produce a good correction is that the erroneous collocation is out of vocabulary. These collocations often involve compound words, like *man-hours* or *carefully-nurturing*, or infrequent expressions, like *copy phenomena*, which do not appear in the FBIS parallel corpus. We expect that this problem can be reduced by using larger parallel corpora for paraphrase extraction.

Near miss (18/100) The second largest category

Model	Rank = 1	Rank $\leq$ 2	Rank $\leq$ 3	P@1	P@2	P@3	MRR
Baseline	43 ǀ 141	69 ǀ 201	83 ǀ 237	18.40	16.68	15.36	12.13 ǀ 36.60
All	**137 ǀ 245**	**176 ǀ 303**	**204 ǀ 340**	**38.20**	**32.87**	**29.30**	**33.16 ǀ 57.26**

Table 6: Results of human evaluation. Rank and MRR results are shown for the intersection (first value) and union (second value) of human judgments.

Original	it must be clear, concise and unambiguous to *prevent* any off-track
Gold	it must be clear, concise and unambiguous to *avoid* any off-track
All	it must be clear, concise and unambiguous to *avoid* any off-track
	it must be clear, concise and unambiguous to *stop* any off-track
	it must be clear, concise and unambiguous to *block* any off-track
Baseline	*it must be clear, concise and unambiguous to *present* any off-track
	it must be clear, concise and unambiguous to *forestall* any off-track
	*it must be clear, concise and unambiguous to *lock* any off-track

Original	although many may *agree* that public spending on the elderly should be limited . . .
Gold	although many may *argue* that public spending on the elderly should be limited . . .
All	although many may *believe* that public spending on the elderly should be limited . . .
	although many may *think* that public spending on the elderly should be limited . . .
	although many may *accept* that public spending on the elderly should be limited . . .
Baseline	*although many may *agreed* that public spending on the elderly should be limited . . .
	*although many may *hold* that public spending on the elderly should be limited . . .
	*although many may *agrees* that public spending on the elderly should be limited . . .

Table 7: Examples of test sentences with the top 3 answers of the ALL and BASELINE system. An answer judged incorrect by at least one judge is marked with an asterisk (*).

Out of vocabulary	. . . many illegal *copy phenomena (copy phenomena, **copies**)* in china.
	. . . lead to reduced *man-hours (man-hours, **productivity**)* as people fall sick . . .
Near miss	. . . smaller groups of people, sometimes *even (more, **only**)* individual .
	. . . take pre-emptive *actions (activities, **measures**)* . . .
Function/auxiliary words	. . . entertainment an elderly person can *have (be, **enjoy**)* .
	. . . and the security issue is solved *also (and, **too**)*
Discourse specific	. . . make other countries respect and fear *you (<question mark>, **a country**)*
	. . . will contribute nothing to the *accident (explosion, **problem**)* .
Spelling errors	this *incidence (rate, **incident**)* had also resulted in 4 fatalities . . .
	refrigerator did not *compromise (yield, **comprise**)* of any moving parts . . .
Word sense	. . . refers to the desire or shortage of a *good (better, **commodity**)* and . . .
	. . . members are always from different *majors (major league, **specialties**)*
Preposition constructions	. . . can be an area worth *investing (investing, **investing in**)*
	. . . in spending their *resources (resources, **resources on**)*
Others	this might *redirect (make sound, **reduce**)* foreign investments . . .
	. . . a trading hub since *british 's (british 's, **british**)* rule.

Table 8: Examples of sentences without valid corrections by the ALL model. The top-1 suggestion of the system and the gold answer (in bold) are shown in parenthesis.

consists of instances where the system barely misses the gold standard answer. This includes cases where the extracted L1-paraphrases do not contain the exact phrase required, e.g., the paraphrase table contains *even|||only get* when the gold correction was *even* → *only*, or the phrase table actually contains the gold answer but fails to rank it among the top 3 answers. The first problem could be addressed by modifying the phrase extraction heuristic to produce more fine-grained phrase pairs. The second problem requires a better language model. Although our language model is trained on the large English Gigaword corpus, it is not always successful in promoting the correct candidate to the top. The domain mismatch between the newswire domain of Gigaword and student essays could be one reason for this.

Function/auxiliary words (14/100) We observe that collocation errors that involve function words or auxiliary words are not handled very well by our model. Function words and auxiliary words in English lack direct counterparts in Chinese, which is why the word alignments and therefore the extracted phrases for these words contain a high amount of noise. As function words and auxiliaries are essentially a closed set, it might be more promising to build separate models with fixed confusion sets for them.

Discourse specific (14/100) Some of the gold answers are highly specific to the particular discourse that they appear in. As our model corrects collocation errors at the sentence level, such gold answers will be very difficult or impossible to determine correctly. Including more context beyond the sentence level might help to overcome this problem, although it is not easy to integrate this larger context information.

Spelling errors (9/100) Some of the collocation errors are caused by spelling mistakes, e.g., *incidence* instead of *incident*. Although the ALL model includes candidates which are created through edit distance, paraphrase candidates created from the misspelled word can dominate the top 3 ranks, e.g., *rate* and *frequently* are paraphrases of *incidence*. A possible solution would be to perform spell-checking as a separate pre-processing step prior to collocation correction.

Word sense (7/100) Some of the failures of the model can be attributed to ambiguous senses of the collocation phrase. As we do not perform word sense disambiguation in our current work, candidates from other word senses can end up as the top candidates. Including word sense disambiguation into the model might help, although accurate word sense disambiguation on noisy learner text may not be easy.

Preposition constructions (6/100) Some of the collocation errors involve preposition constructions, e.g., the student wrote *attend* instead of *attend to*. Because prepositions do not have a direct counterpart in Chinese, the L1-paraphrases do not model their semantics very well. This category is closely related to the function/auxiliary word category. Again, since prepositions are a closed set, it might be more promising to build a separate model for them.

Others (11/100) Other mistakes include collocation errors where the gold answer slightly changed the semantics of the target word, e.g., *redirect potential foreign investments* → *reduce potential foreign investments*, active-passive alternation (*enhanced economics* → *was economical*), and noun possessive errors (*british 's rule* → *british rule*).

8 Conclusion and Future Work

We have presented a novel approach for correcting collocation errors in written learner text. Our approach exploits the semantic similarity of words in the writer's L1-language based on paraphrases extracted from an L1-English parallel corpus. Our experiments on real-world learner data show that our approach outperforms traditional approaches based on edit distance, homophones, and synonyms by a large margin.

In future work, we plan to extend our system to fully automatic collocation correction that involves both identification and correction of collocation errors.

Acknowledgments

This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore.

References

C. Bannard and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*.

C. Brockett, W. B. Dolan, and M. Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of ACL*.

A. J. Carlson, J. Rosen, and D. Roth. 2001. Scaling up context-sensitive text correction. In *Proceedings of IAAI*.

Y. S. Chan and H. T. Ng. 2005. Scaling up word sense disambiguation via parallel texts. In *Proceedings of AAAI*.

Y. C. Chang, J. S. Chang, H. J. Chen, and H. C. Liou. 2008. An automatic collocation writing assistant for Taiwanese EFL learners: A case of corpus-based NLP technology. *Computer Assisted Language Learning*, 21(3):283–299.

D. Dahlmeier and H. T. Ng. 2011. Grammatical error correction with alternating structure optimization. In *Proceedings of ACL*.

M. Farghal and H. Obiedat. 1995. Collocations: A neglected variable in EFL. *International Review of Applied Linguistics*, 33.

C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press, Cambridge,MA.

J. R. Firth. 1957. *Papers in Linguistics 1934-1951*. Oxford University Press, London.

G. Foster, R. Kuhn, and H. Johnson. 2006. Phrasetable smoothing for statistical machine translation. In *Proceedings of EMNLP*.

Y. Futagi, P. Deane, M. Chodorow, and J. Tetreault. 2008. A computational approach to detecting collocation errors in the writing of non-native speakers of English. *Journal of Computer-Assisted Learning*, 21.

M. Gamon. 2010. Using mostly native data to correct errors in learners' writing: A meta-classifier approach. In *Proceedings of HLT-NAACL*.

A. R. Golding and D. Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34:107–130.

A. Haghighi, J. Blitzer, J. DeNero, and D. Klein. 2009. Better word alignments with supervised ITG models. In *Proceedings of ACL-IJCNLP*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL demonstration session*.

J. R. Landis and G. G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1).

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*.

A. L. Liu, D. Wible, and N. L. Tsao. 2009. Automated suggestions for miscollocations. In *Proceedings of the NAACL HLT Workshop on Innovative Use of NLP for Building Educational Applications*.

C. Liu, D. Dahlmeier, and H. T. Ng. 2010a. PEM: a paraphrase evaluation metric exploiting parallel texts. In *Proceedings of EMNLP*.

C. Liu, D. Dahlmeier, and H. T. Ng. 2010b. TESLA: Translation evaluation of sentences with linear-programming-based analysis. In *Proceedings of WMT and MetricsMATR*.

J. K. Low, H. T. Ng, and W. Guo. 2005. A maximum entropy approach to Chinese word segmentation. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*.

N. Madnani and B. J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

D. McCarthy and R. Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*.

J. Meng. 2008. Erroneous collocations caused by language transfer in Chinese EFL writing. *US-China Foreign Language*, 6:57–61.

R. Mitton. 1992. A description of a computer-usable dictionary file based on the Oxford Advanced Learner's Dictionary of Current English.

H. T. Ng and Y. S. Chan. 2007. Semeval-2007 task 11: English lexical sample task via english-chinese parallel text. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)*.

H. T. Ng and J. K. Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of EMNLP*.

H. T. Ng, B. Wang, and Y. S. Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of ACL*.

F. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*.

A. Rozovskaya and D. Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP*.

C. C. Shei and H. Pain. 2000. An ESL writer's collocational aid. *Computer Assisted Language Learning*, 13.

M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of WMT*.

M. Swan and B. Smith. 2001. *Learner English: A Teacher's Guide to Interference and Other Problems*. Cambridge University Press, Cambridge, UK.

J. Tetreault, J. Foster, and M. Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proceedings of ACL*.

D. Wible, C. H. Kuo, N. L. Tsao, A. Liu, and H. L. Lin. 2003. Bootstrapping in a language learning environment. *Journal of Computer-Assisted Learning*, 19.

H. Wu and M. Zhou. 2003. Synonymous collocation extraction using translation information. In *Proceedings of ACL*.

J. C. Wu, Y. C. Chang, T. Mitamura, and J. S. Chang. 2010. Automatic collocation suggestion in academic writing. In *Proceedings of the ACL 2010 Conference Short Papers*.

Z. Zhong and H. T. Ng. 2009. Word sense disambiguation for all words without hard labor. In *Proceedings of IJCAI*.

Class Label Enhancement via Related Instances

Zornitsa Kozareva
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
kozareva@isi.edu

Konstantin Voevodski
Boston University
111 Cummington St.
Boston, MA, 02215
kvodski@bu.edu

Shang-Hua Teng
University of Southern California
941 Bloom Walk, SAL 300
Los Angeles, CA 90089
shanghua@usc.edu

Abstract

Class-instance label propagation algorithms have been successfully used to fuse information from multiple sources in order to enrich a set of unlabeled instances with class labels. Yet, nobody has explored the relationships between the instances themselves to enhance an initial set of class-instance pairs. We propose two graph-theoretic methods (centrality and regularization), which start with a small set of labeled class-instance pairs and use the instance-instance network to extend the class labels to all instances in the network. We carry out a comparative study with state-of-the-art knowledge harvesting algorithm and show that our approach can learn additional class labels while maintaining high accuracy. We conduct a comparative study between class-instance and instance-instance graphs used to propagate the class labels and show that the latter one achieves higher accuracy.

1 Introduction

Many natural language processing applications use and rely on semantic knowledge resources. Since manually built lexical repositories such as WordNet (Fellbaum, 1998) cover a limited amount of knowledge and are tedious to maintain over time, researchers have developed algorithms for automatic knowledge extraction from structured and unstructured texts. There is a substantial body of work on extracting is-a relations (Etzioni et al., 2005; Kozareva et al., 2008), part-of relations (Girju et al., 2003; Pantel and Pennacchiotti, 2006) and general facts (Lin and Pantel, 2001; Davidov and Rappoport, 2009; Jain and Pantel, 2010). The usefulness of the generated resources has been shown to be valuable to information extraction (Riloff and Jones, 1999), question answering (Katz et al., 2003) and textual entailment (Zanzotto et al., 2006) systems.

Among the most common knowledge acquisition approaches are those based on lexical patterns (Hearst, 1992; Etzioni et al., 2005; Kozareva et al., 2008) and clustering (Lin and Pantel, 2002; Davidov and Rappoport, 2008). While clustering can find instances and classes that are not explicitly expressed in text, they often may not generate the granularity needed by the users. In contrast, pattern-based approaches generate highly accurate lists, but they are constraint to the information matched by the pattern and often suffer from recall. (Paşca, 2004; Snow et al., 2006; Kozareva and Hovy, 2010) have shown that complete lists of semantic classes and instances are valuable for the enrichment of existing resources like WordNet and for taxonomy induction. Therefore, researchers have focused on the development of methods that can automatically augment the initially extracted class-instance pairs.

(Pennacchiotti and Pantel, 2009) fused information from pattern-based and distributional systems using an ensemble method and a rich set of features derived from query logs, web-crawl and Wikipedia. (Talukdar et al., 2008) improved class-instance extractions exploring the relationships between the classes and the instances to propagate the initial class-labels to the remaining unlabeled instances. Later on (Talukdar and Pereira, 2010) showed that class-instance extraction with label propagation can be further improved by adding semantic information

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 118–128,
Edinburgh, Scotland, UK, July 27–31, 2011.

in the form of instance-attribute edges derived from independently developed knowledge base. Similarly to (Talukdar et al., 2008) and (Talukdar and Pereira, 2010), we are interested in enriching class-instance extractions with label propagation. However, unlike the previous work, we model the relationships between the instances themselves to propagate the initial set of class labels to the remaining unlabeled instances. To our knowledge, this is the first work to explore the connections between instances for the task of class-label propagation.

Our work addresses the following question: *Is it possible to effectively explore the structure of the text-mined instance-instance networks to enhance an incomplete set of class labels?* Our intuition is that if an instance like *bear* belongs to a semantic class *carnivore*, and the instance *bear* is connected to the instance *fox*, then it is more likely that the unlabeled instance *fox* is also of class *carnivore*. To solve this problem, we propose two graph-based approaches that use the structure of the *instance-instance* graph to propagate the class labels. Our methods are agnostic to the sources of semantic instances and classes. In this work, we carried out experiments with a state-of-the-art instance extraction system and conducted a comparative study between the original and the enhanced class-instance pairs. The results show that this labeling procedure can begin to bridge the gap between the extraction power of the pattern-based approaches and the desired recall by finding class-instance pairs that are not explicitly mentioned in text. The contributions of the paper are as follows:

- We use only the relationships between the instances themselves to propagate class labels.
- We observe how often labels are propagated along the edges of our semantic network, and propose two ways to extend an initial set of class labels to all the instance nodes in the network. The first approach uses a linear system to compute the network centrality relative to the initially labeled instances. The second approach uses a regularization framework with respect to a random walk on the network.
- We evaluate the proposed approaches and show that they discover many new class-instance pairs compared to state-of-the-art knowledge harvesting algorithm, while still maintaining high accuracy.
- We conduct a comparative study between class-instance and instance-instance graphs used to propagate class labels. The experiments show that considering relationships between instances achieves higher accuracy.

The rest of the paper is organized as follows. In Section 2, we review related work. Section 3 describes the Web-based knowledge harvesting algorithm used to extract the instance network and the class-instance pairs necessary for our experimental evaluation. Section 4 describes the two graph-theoretic methods for class label propagation using an instance-instance network. Section 5 shows a comparative study between the proposed graph algorithms and different baselines. We also show a comparison between class-instance and instance-instance graphs used in the label propagation. Finally, we conclude in Section 6.

2 Related Work

In the past decade, we have reached a good understanding on the knowledge harvesting technology from structured (Suchanek et al., 2007) and unstructured text. Researchers have harvested with varying success semantic lexicons (Riloff and Shepherd, 1997) and concept lists (Katz et al., 2003). Many efforts have also focused on the extraction of is-a relations (Hearst, 1992; Paşca, 2004; Etzioni et al., 2005; Paşca, 2007; Kozareva et al., 2008), part-of relations (Girju et al., 2003; Pantel and Pennacchiotti, 2006) and general facts (Etzioni et al., 2005; Davidov and Rappoport, 2009; Jain and Pantel, 2010). Various approaches have been proposed following the patterns of (Hearst, 1992) and clustering (Lin and Pantel, 2002; Davidov and Rappoport, 2008). A substantial body of work has explored issues such as reranking the harvested knowledge using mutual information (Etzioni et al., 2005) and graph algorithms (Hovy et al., 2009), estimating the goodness of text-mining seeds (Vyas et al., 2009), organizing the extracted information (Cafarella et al., 2007a; Cafarella et al., 2007b) and inducing term taxonomies with WordNet (Snow et al., 2006) or starting from scratch (Kozareva and Hovy, 2010).

Since pattern-based approaches tend to be high-precision and low-recall in nature, recently of great interest to the research community is the development of approaches that can increment the recall of the harvested class-instance pairs. (Pennacchiotti and Pantel, 2009) proposed an ensemble semantic framework that mixes distributional and pattern-based systems with a large set of features from a web-crawl, query logs, and Wikipedia. (Talukdar et al., 2008) combined extractions from free text and structured sources using graph-based label propagation algorithm. (Talukdar and Pereira, 2010) conducted a comparative study of graph algorithms and showed that class-instance extraction can be improved using additional information that can be modeled as instance-attribute edges.

Closest to our work is that of (Talukdar et al., 2008; Talukdar and Pereira, 2010) who model class-instance relations to propagate class-labels. Although these algorithms can be applied to other relations (Alfonseca et al., 2010), to our knowledge yet nobody has modeled the connections between the instances themselves for the task of class-label propagation. We propose regularization and centrality graph-theoretic methods, which exploit the instance-instance network and a small set of class-instance pairs to propagate the class-labels to the remaining unlabeled instances. While objectives similar to regularization have been used for class-label propagation, the application of node centrality for this task is also novel. The proposed solutions are intuitive and almost parameter-free (both methods have a single parameter, which is easy to interpret and does not require careful tuning).

3 Knowledge Harvesting from the Web

Our proposed class-label enhancement approaches are agnostic to the sources of semantic instances and classes. Several methods have been developed to harvest instances from the Web (Paşca, 2004; Etzioni et al., 2005; Paşca, 2007; Kozareva et al., 2008) and potentially we can use any of them. In our experiments, we use the doubly-anchored (DAP) method of (Kozareva et al., 2008), because it achieves higher precision than (Etzioni et al., 2005; Paşca, 2007), it is easy to implement and requires minimum supervision (only one seed instance and a lexico-syntactic pattern).

For a given semantic class of interest say *animals*, the algorithm starts with a *seed* example of the class, say *whales*. The *seed* instance is fed into a doubly-anchored pattern "<*semantic-class*> such as <*seed*> and *", which extracts on the position of the * new instances of the semantic class. Then, the newly acquired instances are individually placed on the position of the *seed* in the DAP pattern. The bootstrapping procedure is repeated until no new instances are found. We use the harvested instances to build the instance-instance graph in which the nodes are the learned instances and directed edges like (*whales*,*dolphins*) indicate that the instance *whales* extracted the instance *dolphins*. The edges between the instances are weighted based on the number of times the DAP pattern extracted the instances together.

Different strategies can be employed to acquire semantic classes for each instance. We follow the fully automated approach of (Hovy et al., 2009), which takes the learned instance pairs from DAP and feeds them into the pattern "* such as <*instance*$_1$> and <*instance*$_2$>". The algorithm extracts on the position of the * new semantic classes related to *instance*$_1$. According to (Hovy et al., 2009), the usage of two instances acts as a disambiguator and leads to much more accurate semantic class extraction compared to (Ritter et al., 2009).

4 Methods

We model the output of the instance harvesting algorithm as a directed weighted graph that is given by a set of vertices V and a set of edges E. We use n to denote the number of vertices. A node u corresponds to a learned instance, and an edge $(u, v) \in E$ indicates that the instance v was learned from the instance u using the *DAP* pattern. The weight of the edge $w(u, v)$ specifies the number of times the pair of instances were found by the *DAP* pattern. We define the adjacency matrix of the graph as:

$$A(u,v) = \begin{cases} w(u,v) & \text{if } (u,v) \in E \\ 0 & \text{otherwise.} \end{cases}$$

We use $d_{\text{out}}(u)$ to specify the out-degree of u: $d_{\text{out}}(u) = \sum_{(u,v)\in E} w(u,v)$, and $d_{\text{in}}(v)$ to specify the in-degree of v: $d_{\text{in}}(v) = \sum_{(u,v)\in E} w(u,v)$.

We represent the initial set of instances L that are believed to belong to class C (the set of *labeled* instances) by a row vector $l \in \{0,1\}^n$, where $l(u) = 1$ if $u \in L$. Our objective is to compute a vector $\hat{l}$ where $\hat{l}(u)$ is proportional to how likely it is that u belongs to C. We write all vectors as row vectors, and use $\vec{c}$ to denote a 1 by n *constant* vector such that $\vec{c}(u) = c$ for all $u \in V$.

4.1 Personalized Centrality

Our first approach is based on the intuition that if $u \in C$ and $(u,v) \in E$, then it is more likely that $v \in C$. Moreover, the larger the weight of the edge $w(u,v)$, the more likely it is that $v \in C$. When we extend this intuition to all the in-neighbors, we say that the score of each node is proportional to the sum of the scores of its in-neighbors scaled by the edge weights: $\hat{l}(v) = \alpha \sum_{(u,v)\in E} \hat{l}(u) w(u,v)$. We can verify that the vector $\hat{l}$ must then satisfy $\hat{l} = \alpha \hat{l} A$, so it is an eigenvector of the adjacency matrix of the graph with an eigenvalue of α.

However, this formulation is insufficient because even though it captures our intuition that the nodes get their scores from their in-neighbors, we are still ignoring the initial scores of the nodes. A way to take the initial scores into consideration is to compute the following steady-state equation:

$$\hat{l} = l + \alpha \cdot \hat{l} A. \tag{1}$$

Equation 1 specifies that the score $\hat{l}(u)$ of each node u is the sum of its initial score $l(u)$ and the weighted sum of the scores of its neighbors, which is scaled by α. This equation is known as α-centrality, which was first introduced by (Bonacich and Lloyd, 2001). The α parameter controls how much the score of each node depends on the scores of its neighbors. When $\alpha = 0$ the score of each node is equivalent to its initial score, and does not depend on the scores of its neighbors at all.

Alternately, we can think of the vector $\hat{l}$ as the fixed-point of the process in which in each iteration some node v updates its score $\tilde{l}(v)$ by setting $\tilde{l}(v) = l(v) + \alpha \sum_{(u,v)\in E} w(u,v) \tilde{l}(u)$.

Solving Equation 1 we can see that $\hat{l} = l(I - \alpha A)^{-1}$, where I is the identity matrix of size n. The solution is also closely related to the following expression, which is known as a Katz score (Katz, 1953):

$$s \sum_{t=1}^{\infty} \alpha^t A^t.$$

We can verify that $A^t(u,v)$ gives the number of paths of length t between u and v. Katz proposed using the above expression with the starting vector $s = \vec{1}$ to measure centrality in a network. Therefore, the score of node v is given by the number of paths from u to v for all $u \in V$, with longer paths given less weight based on the value of α. The method proposed here measures a similar quantity with a non-uniform starting vector. To show the relationship between the two measures we use the identity that $\sum_{t=1}^{\infty} \alpha^t A^t = (I - \alpha A)^{-1} - I$. It is easy to see that

$$\begin{aligned} \hat{l} &= l(I - \alpha A)^{-1} \\ &= l(\textstyle\sum_{t=1}^{\infty} \alpha^t A^t + I) \\ &= l \textstyle\sum_{t=1}^{\infty} \alpha^t A^t + l \\ &= l \textstyle\sum_{t=0}^{\infty} \alpha^t A^t. \end{aligned} \tag{2}$$

Equation 2 shows that $\hat{l}(v)$ is given by the number of paths from u to v for all $u \in L$ (the initial labeled set). Using a larger value of α corresponds to giving more weight to paths of longer length. The summation $\sum_{t=0}^{\infty} \alpha^t A^t$ converges as long as $|\alpha| < 1/\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of A. Therefore, we can only consider values of α in this range.

4.2 Regularization Using Random Walks

Our second approach constrains $\hat{l}$ to be as consistent or *smooth* as possible with respect to the structure of the graph. The simplest way to express this is to require that for each edge $(u,v) \in E$, the scores of the endpoints $\hat{l}(u)$ and $\hat{l}(v)$ must be as similar as possible. Moreover, the greater the weight of the edge $w(u,v)$ the more important it is for the scores to match. Using this intuition we can define the following optimization problem:

$$\text{argmin}_{\hat{l}\in\{0,1\}^n} \sum_{(u,v)\in E} (\hat{l}(u) - \hat{l}(v))^2.$$

Setting $\hat{l} = \vec{0}$ or $\hat{l} = \vec{1}$ clearly optimizes this function, but does not give a meaningful solution. However, we can additionally constrain $\hat{l}$ by requiring that the initial labels cannot be modified, or more generally penalizing the discrepancy between $\hat{l}(u)$ and $l(u)$ for $u \in L$. The methods of (Talukdar and Pereira, 2010) optimize objective functions of this type.

Unlike the work of (Talukdar and Pereira, 2010), here we use an objective function that considers smoothness with respect to a *random walk* on the graph. Performing a random walk allows us to take more of the graph structure into account. For example, if nodes u and v are part of the same cluster then it is likely that the edge (u, v) is heavily traversed during the random walk, and should have a lot of probability in the stationary distribution of the walk. Simply considering the weight of the edge $w(u, v)$ gives us no such information. Therefore if our objective function requires the scores to be consistent with respect to the stationary probability of the edges in the random walk, we can compute scores that are consistent with the *clustering structure* of the graph.

Our semantic network is not strongly connected, so we must make some modifications to the random walk to ensure that it has a stationary distribution. Section 4.2.1 describes our random walk and how we compute the transition probability matrix P and its stationary probability distribution π. The definition of our objective function and the description of how it is optimized is given in Section 4.2.2.

4.2.1 Teleporting Random Walk

Formally, a random walk is a process where at each step we move from some node to one of its neighbors. The transition probabilities are given by edge weights, therefore the transition probability matrix W is the normalized adjacency matrix where each row sums to one:

$$W = D^{-1}A.$$

Here the D matrix is the degree matrix, which is a diagonal matrix given by

$$D(u, v) = \begin{cases} d_{\text{out}}(u) & \text{if } u = v \\ 0 & \text{otherwise.} \end{cases}$$

In our semantic network some nodes have no out-neighbors, so in order to compute W we first add a self-loop to any such node. In addition, we modify the random walk to *reset* at each step with nonzero probability β to ensure that it has a steady-state probability distribution. When the walk resets it jumps or *teleports* to any node in the graph with equal probability. The transition probability matrix of this process is given by

$$P = \beta K + (1 - \beta)W,$$

where K is an n by n matrix given by $K(u, v) = \frac{1}{n}$ for all $u, v \in V$. The stationary distribution π must satisfy $\pi = \pi P$. Equivalently π can be viewed as a solution to the following PageRank equation:

$$\pi = \beta s + (1 - \beta)\pi W.$$

Here the *starting* vector $s = \frac{1}{n}\vec{1}$ gives the probability distribution for where the walk transitions when it resets. In our computations we use a jump probability $\beta = 0.15$, which is standard for computations of PageRank. The stationary distribution π can be computed by either solving the PageRank equation or computing the eigenvector of P corresponding to the eigenvalue of 1.

4.2.2 Regularization

(Zhou et al., 2005) propose the following function to measure the smoothness of $\hat{l}$ with respect to the stationary distribution of the random walk:

$$\Omega(\hat{l}) = \frac{1}{2} \sum_{(u,v)\in E} \pi(u)P(u, v)\left(\frac{\hat{l}(u)}{\sqrt{\pi(u)}} - \frac{\hat{l}(v)}{\sqrt{\pi(v)}}\right)^2.$$

Here $\pi(u)P(u, v)$ gives the steady-state probability of traversing the edge (u, v), and $\pi(u)$ and $\pi(v)$ specify how much probability u and v have in the stationary distribution π. Zhou et al. point out that using this function gives better results than smoothness with respect to the edge weights, which can be formulated by replacing $\pi(u)p(u, v)$ with $w(u, v)$, and replacing $\pi(u)$ and $\pi(v)$ with $d_{\text{out}}(u)$ and $d_{\text{in}}(v)$, respectively. This observation is consistent with our intuition that considering a random walk takes more of the graph structure into account.

In addition to minimizing $\Omega(\hat{l})$, we also want $\hat{l}$ to be as close as possible to l, which gives the following optimization problem:

$$\text{argmin}_{\hat{l}\in\mathbb{R}^n}\{\Omega(\hat{y}) + \mu||\hat{l} - l||^2\}. \quad (3)$$

Here the $\mu > 0$ parameter specifies the tradeoff between the two terms: using a larger μ corresponds to placing more emphasis on agreement with the initial labels. (Zhou et al., 2005) show that this objective is optimized by computing

$$\hat{l} = (I - \gamma\Theta)^{-1}l, \quad (4)$$

where $\Theta = (\Pi^{1/2}P\Pi^{-1/2} + \Pi^{-1/2}P\Pi^{1/2})/2$, and $\gamma = 1/(1 + \mu)$. Π is a diagonal matrix given by

$$\Pi(u,v) = \begin{cases} \pi(u) & \text{if } u = v \\ 0 & \text{otherwise.} \end{cases}$$

Zhou et al. propose this approach for semi-supervised learning of labels on the graph, given an initial vector l such that $l(u) = 1$ if vertex u has the label, $l(u) = -1$ if u does not have the label, and $l(u) = 0$ if the vertex is unlabeled. They propose taking the sign of $\hat{l}(u)$ to classify u as positive or negative. Using our labeling procedure we do not have any negative examples, so our initial vector l is non-negative, resulting in a non-negative vector $\hat{l}$. This is not a problem because we can still interpret $\hat{l}(u)$ to be proportional to how likely it is that u has the label. Rather than trying different settings of μ, we directly vary γ, with a smaller γ placing more emphasis on agreement with initial labels.

5 Experimental Evaluation

5.1 Data Collection

For our experimental study, we select three widely used domains in the harvesting community (Etzioni et al., 2005; Paşca, 2007; Hovy et al., 2009; Kozareva and Hovy, 2010): *animals* and *vehicles*. For each domain we randomly selected different semantic classes, which resulted in 20 classes altogether. To generate the instance-instance semantic network, we use the harvesting procedure described in Section 3. For example, to learn instances associated with animals, we instantiate the bootstrapping algorithm with the semantic class *animals*, the seed instance *bears* and the pattern "*animals* such as *bears* and *"". We submitted the pattern as queries to Yahoo!Boss and collected new instances. We ranked the instances following (Kozareva et al., 2008) which resulted in 397 animal, 4471 plant and 1425 vehicle instances. Table 1 shows the number of nodes (instances) and directed edges for the constructed semantic networks.

class	#instances	#directed-edges
animals	397	2812
vehicles	1425	3191

Table 1: *Nodes & Edges in the Instance Network.*

Next, we use the harvested instances to automatically learn the semantic classes associated with them. For example, *bears* and *wolves* are *animals* but also *mammals*, *predators*, *vertebrates* among others. The obtained class harvesting results are shown in Table 2. We indicate with *Inst(Hovy et al., 2009)* the number of instances in the semantic network that discovered the class during the pattern-based harvesting, and with *InstInWordNet* the number of instances in the semantic network belonging to the class according to WordNet.

ClassName	*Inst(Hovy et al., 2009)*	*InstInWordNet*
arthropods	12	50
carnivores	24	57
chordates	2	313
eutherians	3	193
insects	5	29
invertebrates	53	84
mammals	114	205
reptile	5	22
ruminants	14	34
ungulates	16	66
crafts	24	68
motor vehicles	27	127
self-propelled vehicles	36	145
vessels	11	36
wheeled vehicles	54	190

Table 2: *Learned & Gold Standard Class-Instances.*

We can see that the pattern-based approach of (Hovy et al., 2009) does not recover a lot of the class-instance relations present in WordNet. Because of this gap between the actual and the harvested class-instance pairs arises the objective of our work, which is to explore the relationships between the instances to propagate the initially learned class labels to the remaining unlabeled instances. To evaluate the performance of our approach, we use as a gold standard the WordNet class-instance mappings.

5.2 Testing Our Approach

Our approach is based on the intuition that given a labeled instance u of class C, and an instance v in our network, if there is an edge (u, v) then it is more likely that v has the label C as well. For example, if the instance *bears* is of class *vertebrates* and there is an edge between the instances *bears* and *wolves*, then it is likely that *wolves* are also *vertebrates*. Before proceeding with the instance-instance class-label propagation algorithms, first we study whether this intuition is correct.

Individually for each class label C, we construct a set T_C that contains all instances in the network belonging to C according to WordNet. Then we compute the probability that v belongs to C in WordNet

given that (u, v) is an edge in the instance network and u belongs to C in WordNet: $Pr_h = \Pr[v \in T_C \mid (u, v) \in E \text{ and } u \in T_C]$. We compare this to the background probability $Pr_b = \Pr[v \in T_C \mid u, v \in V \text{ and } u \in T_C]$, which gives the probability that v belongs to C in WordNet if it is chosen at random. In other words, if $Pr_h = 1$, this means that whenever u has the label C and (u, v) is an edge, then v is always labeled with C. If indeed this is the case, then a good classifier can simply take the initial set L and extend the labels to all nodes reachable from L in the semantic network. The larger the difference between Pr_h and Pr_b, the more information the links of the instance network carry for the task of label propagation. Table 3 shows the Pr_h and Pr_b values for each class.

CLASS	Pr_h	Pr_b
arthropods	.46	.12
carnivores	.49	.14
chordates	.95	.80
eutherians	.80	.49
insects	.31	.07
invertebrates	.74	.21
mammals	.82	.52
reptile	.27	.05
ruminants	.39	.08
ungulates	.60	.16
crafts	.07	.05
motor vehicles	.10	.09
self-propelled vehicles	.11	.10
vessels	.08	.02
wheeled vehicles	.13	.13

Table 3: *Learned & Gold Standard Class-Instances.*

This study verifies our intuition that using the relationships between the instances to extend a class label to the remaining unlabeled nodes is an effective approach to enhancing an incomplete set of initial labels.

5.3 Comparative Study

The objective of our work is given a set of initially labeled nodes L, to assign to each node a score that indicates how likely it is to belong to L. The simplest way to do this using the edges of the instance network is to say that a node that has more in-neighbors that have a certain label is more likely to have this label. We define the *in-neighbor* score $i(v)$ of a node v as $i(v) = |\{u \in V | (u, v) \in E \text{ and } u \in L\}|$. We expect that the higher the *in-neighbor* score of v, the more likely it is that v has the label L. The *personalized centrality* method that we proposed generalizes this intuition to indirect neighbors (see Methods). Our *regularization using random walks* technique further explores the link structure of the instance network by considering a random walk on it (see Methods). We compare our approaches with a method that labels nodes at *random*. The expected accuracy for class C is given by $\frac{|T_C|}{n}$, where n is the number of nodes in the network, and T_C is the set containing all nodes that belong to C according to WordNet. In other words, given that there are 84 nodes in the network that are classified as *invertebrate* according to WordNet, and there are 397 nodes in total, if we choose any number of nodes at random our expected accuracy is 21%.

We evaluate the performance of our approaches against the WordNet gold standard and show the obtained results in Tables 4 and 5.

Invertebrates				
rank	centrality	regularization	in-neighbor	random
5	1.0	1.0	.80	.21
10	1.0	1.0	.70	.21
20	.95	1.0	.75	.21
50	.96	.98	.76	.21
100	.69	.73	.67	.21
Mammals				
rank	centrality	regularization	in-neighbor	random
5	.80	1.0	.80	.52
10	.90	1.0	.90	.52
20	.95	.95	.85	.52
50	.86	.96	.80	.52
100	.92	.92	.76	.52
Carnivores				
rank	centrality	regularization	in-neighbor	random
5	1.0	1.0	.80	.14
10	.80	.80	.60	.14
20	.80	.85	.55	.14
50	.50	.68	.48	.14
100	.41	.44	.41	.14

Table 4: *Accuracy @ Different Ranks.*

Table 4 shows the accuracy at rank R calculated as the number of correctly labeled instances with class C at rank R divided by the total number of instances with class C at rank R. Due to space limitation, we show detailed ranking only for three of the classes. We can see that using the semantic network significantly enhances our ability to learn class labels. Even the simple *in-neighbor* method produces results that are very significant compared to chance. Our *centrality* and *regularization* techniques further explore the structure of the semantic network to give

better predictions.

Table 5 shows the accuracy of the class label propagation algorithms for each class. For each class we consider the top k ranked nodes, where k is the number of instances that belong to this class according to WordNet. For example, the accuracy of centrality for *carnivores* is 80% showing that from the top 57 ranked animal instances, 80% belong to *carnivores*. In the final column we also report the performance of a label propagation algorithm that uses class-instance graph instead of an instance-instance graph. To build the graph we remove the edges between the instances and keep the class-instance mappings discovered by the harvesting algorithm of (Hovy et al., 2009). We use the modified adsorption algorithm (MAD) of (Talukdar et al., 2008), which is freely available from the Junto toolkit[1]. To rank the instances for each class label produced by Junto, we use the computed label scores as a ranking criteria and measure accuracy similarly to centrality and regularization.

class	Centrality	Regular.	Rand	MAD
arthropods	.50	.60	.12	.56
carnivores	.80	.85	.14	.44
chordates	.81	.83	.80	.79
eutherians	.54	.60	.49	.60
insects	.38	.52	.07	.17
invertebrates	.94	.96	.21	.64
mammals	.82	.90	.52	.63
reptile	.45	.55	.05	.14
ruminants	.41	.44	.08	.41
ungulates	.44	.61	.16	.32
crafts	.47	.56	.05	.35
motor vehicle	.45	.48	.09	.24
self-propelled vehicle	.49	.47	.10	.27
vessel	.33	.39	.02	.31
wheeled vehicle	.51	.52	.13	.33

Table 5: *Comparative Study.*

The obtained results show that for almost all cases the methods that use the structure of the instance network significantly outperform predictions that use the class-instance graph. This indicates that we can indeed learn a lot form the instance-instance relationships by exploring the structure of the instance network. Among all approaches *regularization* achieves the best results. We believe that regularization works well because it considers a random walk on the semantic graph, and within-cluster edges are traversed more often in a random walk. The regularization technique computes scores that are consistent with the clustering structure of the graph by requiring that the endpoints of highly traversed edges, which are likely in the same cluster, have similar scores (see Methods). Overall, *regularization* enhanced the original output generated by the pattern-based knowledge harvesting approach of (Hovy et al., 2009) with 1219 new class-instance pairs (75% additional information) while maintaining 61.87% accuracy.

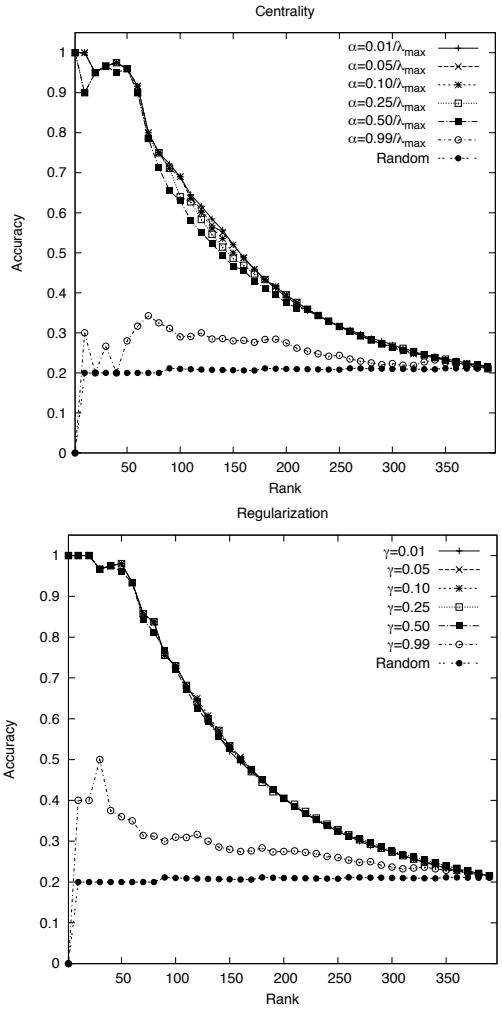


Figure 1: Parameter Tuning For **Invertebrates**.

5.4 Parameter Tuning

Both of our *centrality* and *regularization* methods have a single tunable parameter. For *centrality* the parameter α controls how much the label of each node depends on the labels of its neighbors in the

[1] http://code.google.com/p/junto/

graph. The values range from 0 to $1/\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of the adjacency matrix of the semantic network. When $\alpha = 0$ the label of each node is equivalent to its initial label, while higher values of α give more weight to the labels of nodes that are further away.

For *regularization* the parameter γ controls how much emphasis is placed on the agreement between the initial and learned labels. The values of γ are between 0 and 1. Smaller values require that the learned labels be more consistent with the original labels. When $\gamma = 0$ the learned labels will exactly match the original labels.

For each method we try several parameter settings and show the results in Figure 1 for the propagation of the class label *invertebrate*. We can see that both methods are quite insensitive to the parameter settings, unless we choose very extreme values that ignore the original labels.

5.5 Effect of number of labeled class-instances

We also study how the quality of the results is affected by the number of initial class-instance pairs used by our propagation methods. We conduct experiments using only 25%, 50%, 75% and 100% of the initial class-instance pairs learned by (Hovy et al., 2009). Figure 2 shows the results for the label propagation of the class *invertebrate*.

The performance of our methods significantly improves when we incorporate more labels. Still, if we are less concerned with recall and want to find small sets of nodes with very high accuracy, the number of initial labels is less important. For example, starting with only 13 labeled nodes we can still achieve 100% accuracy for the top 30 nodes using *regularization*, and 96% accuracy for the top 25 nodes using *centrality*.

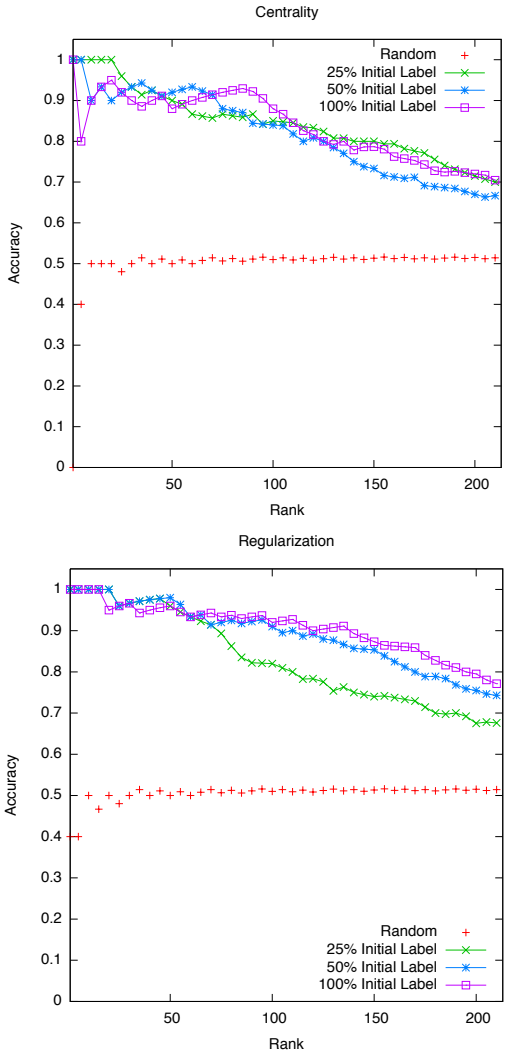


Figure 2: Effect of Number of Initial Class-Instance Pairs for **Invertebrates**.

6 Conclusions

In this paper we proposed a centrality and regularization graph-theoretic methods that explore the relationships between the instances themselves to effectively extend a small set of class-instance labels to all instances in a semantic network. The proposed approaches are intuitive and almost parameter-free. We conducted a series of experiments in which we compared the effectiveness of the centrality and regularization methods to learn new labels for the unlabeled instances. We showed that the enhanced class labels improve the original output generated by the pattern-based knowledge harvesting approach of (Hovy et al., 2009). Finally, we have studied the impact of the class-instance and instance-instance graphs for the class-label propagation task. The latter approach has shown to produce much more accurate results. In the future, we want to apply our approach to Web-based taxonomy induction, which according to (Kozareva and Hovy, 2010) is stifled due to the lacking relations between the instances and the classes, and the classes themselves. The proposed methods can be also applied to enhance fact

farms (Jain and Pantel, 2010).

Acknowledgments

We acknowledge the support of DARPA contract number FA8750-09-C-3705 and NSF grant IIS-0429360. We would like to thank the three anonymous reviewers for their useful comments and suggestions and Partha Talukdar for the discussions on modified adsorption.

References

Enrique Alfonseca, Marius Pasca, and Enrique Robledo-Arnuncio. 2010. Acquisition of instance attributes via labeled and related instances. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 58–65.

Phillip Bonacich and Paulette Lloyd. 2001. *Social Networks*, 23(3):191–201.

Michael J. Cafarella, Christopher R, Dan Suciu, Oren Etzioni, and Michele Banko. 2007a. Structured querying of web text: A technical challenge. In *in CIDR*.

Michael J. Cafarella, Dan Suciu, and Oren Etzioni. 2007b. Navigating extracted data with schema discovery. In *Tenth International Workshop on the Web and Databases, WebDB 2007WebDB*.

Dmitry Davidov and Ari Rappoport. 2008. Classification of semantic relationships between nominals using pattern clusters. In *Proceedings of ACL-08: HLT*, pages 227–235, June.

Dmitry Davidov and Ari Rappoport. 2009. Geo-mining: Discovery of road and transport networks using directional patterns. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP-09*, pages 267–275.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 1–8.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545.

Eduard Hovy, Zornitsa Kozareva, and Ellen Riloff. 2009. Toward completeness in concept extraction and classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 948–957.

Alpa Jain and Patrick Pantel. 2010. Factrank: Random walks on a web of facts. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 501–509.

Boris Katz, Jimmy Lin, Daniel Loreto, Wesley Hildebrandt, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Gregory Marton, and Federico Mora. 2003. Integrating web-based and corpus-based techniques for question answering. In *Proceedings of the twelfth text retrieval conference (TREC)*, pages 426–435.

Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1110–1118.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics ACL-08: HLT*, pages 1048–1056.

Dekang Lin and Patrick Pantel. 2001. Dirt - discovery of inference rules from text. In *In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proc. of the 19th international conference on Computational linguistics*, pages 1–7.

Marius Paşca. 2004. Acquisition of categorized named entities for web search. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145.

Marius Paşca. 2007. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *Proceedings of the 16th international conference on World Wide Web*, pages 101–110.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 113–120.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural*

Language Processing: Volume 1 - Volume 1, EMNLP '09, pages 238–247.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI '99/IAAI '99: Proceedings of the Sixteenth National Conference on Artificial intelligence.*

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of the Empirical Methods for Natural Language Processing*, pages 117–124.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of the AAAI Spring Symposium on Learning by Reading and Learning to Read*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 801–808.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Partha Pratim Talukdar and Fernando Pereira. 2010. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1473–1481.

Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 582–590.

Vishnu Vyas, Patrick Pantel, and Eric Crestan. 2009. Helping editors choose better seed sets for entity set expansion. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 225–234.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Maria Teresa Pazienza. 2006. Discovering asymmetric entailment relations between verbs using selectional preferences. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 849–856.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2005. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 1036–1043.

A Joint Model for Extended Semantic Role Labeling

Vivek Srikumar and Dan Roth
University of Illinois, Urbana-Champaign
Urbana, IL 61801
{vsrikum2, danr}@illinois.edu

Abstract

This paper presents a model that extends semantic role labeling. Existing approaches independently analyze relations expressed by verb predicates or those expressed as nominalizations. However, sentences express relations via other linguistic phenomena as well. Furthermore, these phenomena interact with each other, thus restricting the structures they articulate. In this paper, we use this intuition to define a joint inference model that captures the inter-dependencies between verb semantic role labeling and relations expressed using prepositions. The scarcity of jointly labeled data presents a crucial technical challenge for learning a joint model. The key strength of our model is that we use existing structure predictors as black boxes. By enforcing consistency constraints between their predictions, we show improvements in the performance of both tasks without retraining the individual models.

1 Introduction

The identification of semantic relations between sentence constituents has been an important task in NLP research. It finds applications in various natural language understanding tasks that require complex inference going beyond the surface representation. In the literature, semantic role extraction has been studied mostly in the context of verb predicates, using the Propbank annotation of Palmer et al. (2005), and also for nominal predicates, using the Nombank corpus of Meyers et al. (2004).

However, sentences express semantic relations through other linguistic phenomena. For example, consider the following sentence:

(1) The field goal by Brien changed the game in the fourth quarter.

Verb centered semantic role labeling would identify the arguments of the predicate *change* as (a) *The field goal by Brien* (A0, the causer of the change), (b) *the game* (A1, the thing changing), and (c) *in the fourth quarter* (temporal modifier). However, this does not tell us that the scorer of the field goal was Brien, which is expressed by the preposition *by*. Also, note that the *in* indicates a temporal relation, which overlaps with the verb's analysis.

In this paper, we propose an extension of the standard semantic role labeling task to include relations expressed by lexical items other than verbs and nominalizations. Further, we argue that there are interactions between the different phenomena which suggest that there is a benefit in studying them together. However, one key challenge is that large jointly labeled corpora do not exist. This motivates the need for novel learning and inference schemes that address the data problem and can still benefit from the interactions among the phenomena.

This paper has two main contributions.

1. From the machine learning standpoint, we propose a joint inference scheme to combine *existing* structure predictors for multiple linguistic phenomena. We do so using hard constraints that involve only the labels of the phenomena. The strength of our model is that it is easily

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 129–139,
Edinburgh, Scotland, UK, July 27–31, 2011.

extensible, since adding new phenomena does not require fully retraining the joint model from scratch. Furthermore, our approach minimizes the need for extensive jointly labeled corpora and, instead, uses existing predictors as black boxes.

2. From an NLP perspective, we motivate the extension of semantic role labeling beyond verbs and nominalizations. We instantiate our joint model for the case of extracting preposition and verb relations together. Our model uses existing systems that identify verb semantic roles and preposition object roles and jointly predicts the output of the two systems in the presence of linguistic constraints that enforce coherence between the predictions. We show that using constraints to combine models improves the performance on both tasks. Furthermore, since the constraints depend only on the labels of the two tasks and not on any specific dataset, our experiments also demonstrate that enforcing them allows for better domain adaptation.

The rest of the paper is organized as follows: We motivate the need for extending semantic role labeling and the necessity for joint inference in Section 2. In Section 3, we describe the component verb SRL and preposition role systems. The global model is defined in Section 4. Section 5 provides details on the coherence constraints we use and demonstrates the effectiveness of the joint model through experiments. Section 6 discusses our approach in comparison to existing work and Section 7 provides concluding remarks.

2 Problem Definition and Motivation

Semantic Role Labeling has been extensively studied in the context of verbs and nominalizations. While this analysis is crucial to understanding a sentence, it is clear that in many natural language sentences, information is conveyed via other lexical items. Consider, for example, the following sentences:

(2) Einstein's theory of relativity changed physics.

(3) The plays of Shakespeare are widely read.

(4) The bus, which was heading for Nairobi in Kenya, crashed in the Kabale district of Uganda.

The examples contain information that cannot be captured by analyzing the verbs and the nominalizations. In sentence (2), the possessive form tells us that the *theory of relativity* was discovered by *Einstein*. Furthermore, the *theory* is on the subject of *relativity*. The usage of the preposition *of* is different in sentence (3), where it indicates a creator-creation relationship. In the last sentence, the same preposition tells us that the Kabale district is located in Uganda. Prepositions, compound nouns, possessives, adjectival forms and punctuation marks often express relations, the identification of which is crucial for text understanding tasks like recognizing textual entailment, paraphrasing and question answering.

The relations expressed by different linguistic phenomena often overlap. For example, consider the following sentence:

(5) Construction of the library began in 1968.

The relation expressed by the nominalization *construction* recognizes *the library* as the argument of the predicate *construct*. However, the same analysis can also be obtained by identifying the sense of the preposition *of*, which tells us that the subject of the preposition is a nominalization of the underlying verb. A similar redundancy can be observed with analyses of the verb *began* and the preposition *in*. The above example motivates the following key intuition: *The correct interpretation of a sentence is the one that gives a consistent analysis across all the linguistic phenomena expressed in it.*

An inference mechanism that simultaneously predicts the structure for different phenomena should account for consistency between the phenomena. A model designed to address this has the following desiderata:

1. It should account for the dependencies between phenomena.

2. It should be extensible to allow easy addition of new linguistic phenomena.

3. It should be able to leverage existing state-of-the-art models with minimal use of jointly labeled data, which is expensive to obtain.

Systems that are trained on each task independently do not account for the interplay between them. One approach for tackling this is to define pipelines, where the predictions for one of the tasks acts as the input for another. However, a pipeline does not capture the two-way dependency between the tasks. Training a fully joint model from scratch is also unrealistic because it requires text that is annotated with all the tasks, thus making joint training implausible from a learning theoretic perspective (See Punyakanok et al. (2005) for a discussion about the learning theoretic requirements of joint training.)

3 Tasks and Individual Systems

Before defining our proposed model that captures the requirements listed in the previous section, we introduce the tasks we consider and their independently trained systems that we improve using the joint system. Though the model proposed here is general and can be extended to several linguistic phenomena, in this paper, we focus on relations expressed by verbs and prepositions. This section describes the tasks, the data sets we used for our experiments and the current state-of-the-art systems for these tasks.

We use the following sentence as our running example to illustrate the phenomena: *The company calculated the price trends on the major stock markets on Monday.*

3.1 Preposition Relations

Prepositions indicate a relation between the attachment point of the preposition and its object. As we have seen, the same preposition can indicate different types of relations. In the literature, the polysemy of prepositions is addressed by The Preposition Project[1] of Litkowski and Hargraves (2005), which is a large lexical resource for English that labels prepositions with their sense. This sense inventory formed the basis of the SemEval-2007 task of preposition word sense disambiguation of Litkowski and Hargraves (2007). In our example, the first *on* would be labeled with the sense **8(3)** which identifies the object of the preposition as the topic, while the second instance would be labeled as **17(8)**, which indicates that argument is the day of the occurrence.

The preposition sense inventory, while useful to identify the fine grained distinctions between preposition usage, defines a unique sense label for each preposition by indexing the definitions of the prepositions in the Oxford Dictionary of English. For example, in the phrase *at noon*, the at would be labeled with the sense **2(2)**, while the preposition in *I will see you in an hour* will be labeled **4(3)**. Note that both these (and also the second *on* in our running example) indicate a temporal relation, but are assigned different labels based on the preposition. To counter this problem we collapsed preposition senses that are semantically similar to define a new label space, which we refer to as *Preposition Roles*.

We retrained classifiers for preposition sense for the new label space. Before describing the preposition role dataset, we briefly describe the datasets and the features for the sense problem. The best performing system at the SemEval-2007 shared task of preposition sense disambiguation (Ye and Baldwin (2007)) achieves a mean precision of 69.3% for predicting the fine grained senses. Tratz and Hovy (2009) and Hovy et al. (2010) attained significant improvements in performance using features derived from the preposition's neighbors in the parse tree. We extended the feature set defined in the former for our independent system. Table 1 summarizes the rules for identifying the syntactically related words for each preposition. We used dependencies from the easy-first dependency parser of Goldberg and Elhadad (2010).

For each word extracted from these rules, the features include the word itself, its lemma, the POS tag, synonyms and hypernyms of the first WordNet sense and an indicator for capitalization. These features improved the accuracy of sense identification to 75.1% on the SemEval test set. In addition, we also added the following new features for each word:

1. Indicators for gerunds and nominalizations of verbs.

2. The named entity tag (Person, Location or Organization) associated with a word, if any. We

[1] http://www.clres.com/prepositions.html

Id.	Feature
1.	Head noun/verb that dominates the preposition along with its modifiers
2.	Head noun/verb that is dominated by the preposition along with its modifiers
3.	Subject, negator and object(s) of the immediately dominating verb
4.	Heads of sibling prepositions
5.	Words withing a window of 5 centered at the preposition

Table 1: Features for preposition relation from Tratz and Hovy (2009). These rules were used to identify syntactically related words for each preposition.

used the state-of-the-art named entity tagger of Ratinov and Roth (2009) to label the text.

3. Gazetteer features, which are active if a word is a part of a phrase that belongs to a gazetteer list. We used the gazetteer lists which were used by the NER system. We also used the CBC word clusters of Pantel and Lin (2002) as additional gazetteers and Brown cluster features as used by Ratinov and Roth (2009) and Koo et al. (2008).

Dahlmeier et al. (2009) annotated senses for the prepositions *at, for, in, of, on, to* and *with* in the sections 2-4 and 23 of the Wall Street Journal portion of the Penn Treebank[2]. We trained sense classifiers on both datasets using the Averaged Perceptron algorithm with the one-vs-all scheme using the Learning Based Java framework of Rizzolo and Roth (2010)[3]. Table 2 reports the performance of our sense disambiguation systems for the Treebank prepositions.

As mentioned earlier, we collapsed the sense labels onto the newly defined preposition role labels. Table 3 shows this label set along with frequencies of the labels in the Treebank dataset. According to this labeling scheme, the first *on* in our running example will be labeled TOPIC and the second one will be labeled TEMPORAL[4]. We re-trained the sense disambiguation system to predict preposition roles. When trained on the Treebank data, our system attains an accuracy of 67.82% on Section 23 of the Treebank. We use this system as our independent baseline for preposition role identification.

Train	Test set	
	Treebank Sec. 23	SemEval
Penn Treebank	61.41	38.22
SemEval	47.00	78.25

Table 2: Preposition sense performance. This table reports accuracy of sense prediction on the prepositions that have been annotated for the Penn Treebank dataset.

Role	Train	Test
ACTIVITY	57	23
ATTRIBUTE	119	51
BENEFICIARY	78	17
CAUSE	255	116
CONCOMITANT	156	74
ENDCONDITION	88	66
EXPERIENCER	88	42
INSTRUMENT	37	19
LOCATION	1141	414
MEDIUMOFCOMMUNICATION	39	30
NUMERIC/LEVEL	301	174
OBJECTOFVERB	365	112
OTHER	65	49
PARTWHOLE	485	133
PARTICIPANT/ACCOMPANIER	122	58
PHYSICALSUPPORT	32	18
POSSESSOR	195	56
PROFESSIONALASPECT	24	10
RECIPIENT	150	70
SPECIES	240	58
TEMPORAL	582	270
TOPIC	148	54

Table 3: Preposition role data statistics for the Penn Treebank preposition dataset.

3.2 Verb SRL

The goal of verb Semantic Role Labeling (SRL) is to identify the predicate-argument structure defined by verbs in sentences. The CoNLL Shared Tasks of 2004 and 2005 (See Carreras and Màrquez

[2]This dataset does not annotate all prepositions and restricts itself mainly to prepositions that start a Propbank argument. The data is available at `http://nlp.comp.nus.edu.sg/corpora`

[3]Learning Based Java can be downloaded from `http://cogcomp.cs.illinois.edu`.

[4]The mapping from the preposition senses to the roles defines a new dataset and is available for download at `http://cogcomp.cs.illinois.edu/`.

(2004), Carreras and Màrquez (2005)) studied the identification of the predicate-argument structure of verbs using the PropBank corpus of Palmer et al. (2005). Punyakanok et al. (2008) and Toutanova et al. (2008) used global inference to ensure that the predictions across all arguments of the same predicate are coherent. We re-implemented the system of Punyakanok et al. (2008), which we briefly describe here, to serve as our baseline verb semantic role labeler [5]. We refer the reader to the original paper for further details.

The verb SRL system of Punyakanok et al. (2008) consists of four stages – candidate generation, argument identification, argument classification and inference. The candidate generation stage involves using the heuristic of Xue and Palmer (2004) to generate an over-complete set of argument candidates for each predicate. The identification stage uses a classifier to prune the candidates. In the argument classification step, the candidates that remain after the identification step are assigned scores for the SRL arguments using a multiclass classifier. One of the labels of the classifier is $\emptyset$, which indicates that the candidate is, in fact, not an argument. The inference step produces a combined prediction for all argument candidates of a verb proposition by enforcing global constraints.

The inference enforces the following structural and linguistic constraints: (1) Each candidate can have at most one label. (2) No duplicate core arguments. (3) No overlapping or embedding arguments. (4) Given the predicate, some argument classes are illegal. (5) If a candidate is labeled as an R-arg, then there should be one labeled as arg. (6) If a candidate is labeled as a C-arg, there should be one labeled arg that occurs *before* the C-arg.

Instead of using the identifier to filter candidates for the classifier, in our SRL system, we added the identifier to the global inference and enforced consistency constraints between the identifier and the argument classifier predictions – the identifier should predict that a candidate is an argument if, and only if, the argument classifier does not predict the label $\emptyset$. This change is in keeping with the idea of using joint inference to combine independently learned systems, in this case, the argument identifier and the role classifier. Furthermore, we do not need to explicitly tune the identifier for high recall.

We phrase the inference task as an integer linear program (ILP) following the approach developed in Roth and Yih (2004). Integer linear programs were used by Roth and Yih (2005) to add general constraints for inference with conditional random fields. ILPs have since been used successfully in many NLP applications involving complex structures – Punyakanok et al. (2008) for semantic role labeling, Riedel and Clarke (2006) and Martins et al. (2009) for dependency parsing and several others[6].

Let $v^C_{i,a}$ be the Boolean indicator variable that denotes that the i^{th} argument candidate for a predicate is assigned a label a and let $\Theta^C_{i,a}$ represent the score assigned by the argument classifier for this decision. Similarly, let v^I_i denote the identifier decision for the i^{th} argument candidate of the predicate and Θ^I_i denote its identifier score. Then, the objective of inference is to maximize the total score of the assignment

$$\max_{\mathbf{v}^C,\mathbf{v}^I} \sum_{i,a} \Theta^C_{i,a} v^C_{i,a} + \sum_i \Theta^I_i v^I_i \qquad (1)$$

Here, $\mathbf{v}^C$ and $\mathbf{v}^I$ denote all the argument classifier and identifier variables respectively. This maximization is subject to the constraints described above, which can be transformed to linear (in)equalities. We denote these constraints as $\mathcal{C}^{SRL}$. In addition to $\mathcal{C}^{SRL}$ which were defined by Punyakanok et al. (2008), we also have the constraints linking the predictions of the identifier and classifier:

$$v^C_{v,i,\emptyset} + v^I_{v,i} = 1; \;\; \forall v, i. \qquad (2)$$

Inference in our baseline SRL system is, thus, the maximization of the objective defined in (1) subject to constraints $\mathcal{C}^{SRL}$, the identifier-classifier constraints defined in (2) and the restriction of the variables to take values in $\{0, 1\}$.

To train the classifiers, we used parse trees from the Charniak and Johnson (2005) parser with the

[5]The verb SRL system be downloaded from `http://cogcomp.cs.illinois.edu/page/software`

[6]The primary advantage of using ILP for inference is that this representation enables us to add arbitrary coherence constraints between the phenomena. If the underlying optimization problem itself is tractable, then so is the corresponding integer program. However, other approaches to solve the constrained maximization problem can also be used for inference.

same feature representation as in the original system. We trained the classifiers on the standard Propbank training set using the one-vs-all extension of the average Perceptron algorithm. As with the preposition roles, we implemented our system using Learning Based Java of Rizzolo and Roth (2010). We normalized all classifier scores using the softmax function. Compared to the 76.29% F1 score reported by Punyakanok et al. (2008) using single parse tree predictions from the parser, our system obtained 76.22% F1 score on section 23 of the Penn Treebank.

4 A Joint Model for Verbs and Prepositions

We now introduce our model that captures the needs identified in Section 2. The approach we develop in this paper follows the one proposed by Roth and Yih (2004) of training individual models and combining them at inference time. Our joint model is a Constrained Conditional Model (See Chang et al. (2011)), which allows us to build upon existing learned models using declarative constraints.

We represent our component inference problems as integer linear program instances. As we saw in Section 3.2, the inference for SRL is instantiated as an ILP problem. The problem of predicting preposition roles can be easily transformed into an ILP instance. Let $v^R_{p,r}$ denote the decision variable that encodes the prediction that the preposition p is assigned a role r and let $\Theta^R_{p,r}$ denote its score. Let $\mathbf{v}^R$ denote all the role variables for a sentence. Then role prediction is equivalent to the following maximization problem:

$$\max_{\mathbf{v}^R} \quad \sum_{p,r} \Theta^R_{p,r} \cdot v^R_{p,r} \tag{3}$$

$$\text{subj. to} \quad \sum_{r} v^R_{p,r} = 1, \quad \forall p \tag{4}$$

$$v^R_{p,r} \in \{0,1\}, \quad \forall p, r. \tag{5}$$

In general, let p denote a linguistic structure prediction task of interest and let $\mathcal{P}$ denote all such tasks. Let Z^p denote the set of labels that the parts of the structure associated with phenomenon p can take. For example, for the SRL argument classification component, the parts of the structure are all the candidates that need to be labeled for a given sentence and the set Z^p is the set of all argument labels. For each phenomenon $p \in \mathcal{P}$, we use $\mathbf{v}^p$ to denote its set of inference variables for a given sentence. Each inference variable $v^p_{Z,y} \in \mathbf{v}^p$ corresponds to the prediction that the part y has the label Z in the final structure. Each variable is associated with a score $\Theta^p_{Z,y}$ that is obtained from a learned score predictor. Let $\mathcal{C}^p$ denote the structural constraints that are "local" to the phenomenon. Thus, for verb SRL, these would be the constraints defined in the previous section, and for preposition role, the only local constraint would be the constraint (4) defined above.

The independent inference problem for the phenomenon p is the following integer program:

$$\max_{\mathbf{v}^p} \quad \sum_{Z \in Z^p} \sum_{v^p} v^p_{Z,y} \cdot \Theta^p_{Z,y}, \tag{6}$$

$$\text{subj. to} \quad \mathcal{C}^p(\mathbf{v}^p), \tag{7}$$

$$v^p_{Z,y} \in \{0,1\}, \qquad \forall v^p_{Z,y}. \tag{8}$$

As a technical point, this defines one inference problem per sentence, rather than per predicate as in the verb SRL system of Punyakanok et al. (2008). This simple extension enabled Surdeanu et al. (2007) to study the impact of incorporating cross-predicate constraints for verb SRL. In this work, this extension allows us to incorporate cross-phenomena inference.

4.1 Joint inference

We consider the problem of jointly predicting several phenomena incorporating linguistic knowledge that enforce consistency between the output labels. Suppose p_1 and p_2 are two phenomena. If $z^{p_1}_1$ is a label associated with the former and $z^{p_2}_1, z^{p_2}_2, \cdots$ are labels associated with the latter, we consider constraints of the form

$$z^{p_1}_1 \rightarrow z^{p_2}_1 \vee z^{p_2}_2 \vee \cdots \vee z^{p_2}_n \tag{9}$$

We expand this language of constraints by allowing the specification of pre-conditions for a constraint to apply. This allows us to enforce constraints of the form *"If an argument that starts with the preposition 'at' is labeled* AM-TMP*, then the preposition can be labeled either* NUMERIC/LEVEL *or* TEMPORAL*."* This constraint is universally quantified for

all arguments that satisfy the precondition of starting with the preposition *at*.

Given a first-order constraint in this form and an input sentence, suppose the inference variable $v_1^{p_1}$ is a grounding of $z_1^{p_1}$ and $v_1^{p_2}, v_2^{p_2}, \cdots$ are groundings of the right hand labels such that the preconditions are satisfied, then the constraint can be phrased as the following linear inequality.

$$-v_1^{p_1} + \sum_i v_i^{p_2} \geq 0$$

In the context of the preposition role and verb SRL, we consider constraints between labels for a preposition and SRL argument candidates that begin with that preposition. This restriction forms the precondition for all the joint constraints considered in this paper. Since the joint constraints involve only the labels, they can be derived either manually from the definition of the tasks or using statistical relation learning techniques. In addition to mining constraints of the form (9), we also use manually specified joint constraints. The constraints used in our experiments are described further in Section 5.

In general, let J denote a set of pairwise joint constraints. The joint inference problem can be phrased as that of maximizing the score of the assignment subject to the structural constraints of each phenomenon ($\mathcal{C}^p$) *and* the joint linguistic constraints (J). However, since, the individual tasks were not trained on the same datasets, the scoring functions need not be in the same numeric scale. In our model, each label Z for a phenomenon p is associated with a scoring function $\Theta^p_{Z,y}$ for a part y. To scale the scoring functions, we associate each label with a parameter λ^p_Z. This gives us the following integer linear program for joint inference:

$$\max_{\mathbf{v}} \quad \sum_{p \in \mathcal{P}} \sum_{Z \in Z^p} \lambda^p_Z \left(\sum_{y^p} v^p_{Z,y} \cdot \Theta^p_{Z,y} \right), \quad (10)$$

$$\text{subj. to} \quad \mathcal{C}^p(\mathbf{v}^p), \quad \forall p \in \mathcal{P} \quad (11)$$

$$J(\mathbf{v}), \quad (12)$$

$$v^p_{Z,y} \in \{0, 1\}, \quad \forall v^p_{Z,y}. \quad (13)$$

Here, $\mathbf{v}$ is the vector of inference variables which is obtained by stacking all the inference variables of each phenomena.

For our experiments, we use a cutting plane solver to solve the integer linear program as in Riedel (2009). This allows us to solve the inference problem without explicitly having to instantiate all the joint constraints.

4.2 Learning to rescale the individual systems

Given the individual models and the constraints, we only need to learn the scaling parameters λ^p_Z. Note that the number of scaling parameters is the total number of labels. When we jointly predict verb SRL and preposition role, we have 22 preposition roles (from table 3), one SRL identifier label and 54 SRL argument classifier labels. Thus we learn only 77 parameters for our joint model. This means that we only need a very small dataset that is jointly annotated with all the phenomena.

We use the Structure Perceptron of Collins (2002) to learn the scaling weights. Note that for learning the scaling weights, we need each label to be associated with a real-valued feature. Given an assignment of the inference variables $\mathbf{v}$, the value of the feature corresponding to the label Z of task p is given by the sum of scores of all parts in the structure for p that have been assigned this label, i.e. $\sum_{y^p} v^p_{Z,y} \cdot \Theta^p_{Z,y}$. This feature is computed for the gold and the predicted structures and is used for updating the weights.

5 Experiments

In this section, we describe our experimental setup and evaluate the performance of our approach. The research question addressed by the experiments is the following: *Given independently trained systems for verb SRL and preposition roles, can their performance be improved using joint inference between the two tasks?* To address this, we report the results of the following two experiments:

1. First, we compare the joint system against the baseline systems and with pipelines in both directions. In this setting, both base systems are trained on the Penn Treebank data.

2. Second, we show that using joint inference can provide strong a performance gain even when the underlying systems are trained on different domains.

In all experiments, we report the F1 measure for the verb SRL performance using the CoNLL 2005

evaluation metric and the accuracy for the preposition role labeling task.

5.1 Data and Constraints

For both the verb SRL and preposition roles, we used the first 500 sentences of section 2 of the Penn Treebank corpus to train our scaling parameters. For the first set of experiments, we trained our underlying systems on the rest of the available Penn Treebank training data for each task. For the adaptation experiment, we train the role classifier on the SemEval data (restricted to the same Treebank prepositions). In both cases, we report performance on section 23 of the Treebank.

We mined consistency constraints from the sections 2, 3 and 4 of the Treebank data. As mentioned in Section 4.1, we considered joint constraints relating preposition roles to verb argument candidates that start with the preposition. We identified the following types of constraints: (1) For each preposition, the set of invalid verb arguments and preposition roles. (2) For each preposition role, the set of allowed verb argument labels if the role occurred more than ten times in the data, and (3) For each verb argument, the set of allowed preposition roles, similarly with a support of ten. Note that, while the constraints were obtained from jointly labeled data, the constraints could be written down because they encode linguistic intuition about the labels.

The following is a constraint extracted from the data, which applies to the preposition *with*:

$$
\begin{aligned}
\text{srlarg(A2)} \quad &\rightarrow \quad \text{prep-role(ATTRIBUTE)} \\
&\vee \quad \text{prep-role(CAUSE)} \\
&\vee \quad \text{prep-role(INSTRUMENT)} \\
&\vee \quad \text{prep-role(OBJECTOFVERB)} \\
&\vee \quad \text{prep-role(PARTWHOLE)} \\
&\vee \quad \text{prep-role(PARTICIPANT/ACCOMPAINER)} \\
&\vee \quad \text{prep-role(PROFESSIONALASPECT).}
\end{aligned}
$$

This constraint says that if any candidate that starts with *with* is labeled as an A2, then the preposition can be labeled only with one of the roles on the right hand side.

Some of the mined constraints have negated variables to enforce that a role or an argument label should not be allowed. These can be similarly converted to linear inequalities. See Rizzolo and Roth (2010) for a further discussion about converting logical expressions into linear constraints.

In addition to these constraints that were mined from data, we also enforce the following handwritten constraints: (1) If the role of a verb attached preposition is labeled TEMPORAL, then there should be a verb predicate for which this prepositional phrase is labeled AM-TMP. (2) For verb attached prepositions, if the preposition is labeled with one of ACTIVITY, ENDCONDITION, INSTRUMENT or PROFESSIONALASPECT, there should be at least one predicate for which the corresponding prepositional phrase is not labeled $\emptyset$.

The conversion of the first constraint to a linear inequality is similar to the earlier cases. For each of the roles in the second constraint, let r denote a role variable that assigns the label to some preposition. Suppose there are n SRL candidates across all verb predicates begin with that preposition, and let $s_1, s_2, \cdots, s_n$ denote the SRL variables that assign these candidates to the label $\emptyset$. Then the second constraint corresponds to the following inequality:

$$r + \sum_{i=1}^{n} s_i \leq n$$

5.2 Results of joint learning

First, we compare our approach to the performance of the baseline independent systems and to pipelines in both directions in Table 4. For one pipeline, we added the prediction of the baseline preposition role system as an additional feature to both the identifier and the argument classifier for argument candidates that start with a preposition. Similarly, for the second pipeline, we added the SRL predictions as features for prepositions that were the first word of an SRL argument. In all cases, we performed five-fold cross validation to train the classifiers.

The results show that both pipelines improve performance. This justifies the need for a joint system because the pipeline can improve only one of the tasks. The last line of the table shows that the joint inference system improves upon both the baselines. We achieve this improvement without retraining the underlying models, as done in the case of the pipelines.

On analyzing the output of the systems, we found that the SRL precision improved by 2.75% but the

Setting	SRL (F1)	Preposition Role (Accuracy)
Baseline SRL	76.22	–
Baseline Prep.	–	67.82
Prep. → SRL	**76.84**	–
SRL → Prep.	–	**68.55**
Joint inference	**77.07**	**68.39**

Table 4: Performance of the joint system, compared to the individual systems and the pipelines. All performance measures are reported on Section 23 of the Penn Treebank. The verb SRL systems were trained on sections 2-21, while the preposition role classifiers were trained on sections 2-4. For the joint inference system, the scaling parameters were trained on the first 500 sentences of section 2, which were held out. All the improvements in this table are statistically significant at the 0.05 level.

recall decreased by 0.98%, contributing to the overall F1 improvement. The decrease in recall is due to the joint hard constraints that prohibit certain assignments to the variables which would have otherwise been possible. Note that, for a given sentence, even if the joint constraints affect only a few argument candidates directly, they can alter the labels of the other candidates via the "local" SRL constraints.

Consider the following example of the system output which highlights the effect of the constraints.

(6) Weatherford said market conditions led to the cancellation of the planned exchange.

The independent preposition role system incorrectly identifies the *to* as a LOCATION. The semantic role labeling component identifies the phrase *to the cancellation of the planned exchange* as the A2 of the verb *led*. One of the constraints mined from the data prohibits the label LOCATION for the preposition *to* if the argument it starts is labeled A2. This forces the system to change the preposition label to the correct one, namely ENDCONDITION. Both the independent and the joint systems also label the preposition *of* as OBJECTOFVERB, which indicates that the phrase *the planned exchange* is the object of the deverbal noun *cancellation*.

5.3 Effect of constraints on adaptation

Our second experiment compares the performance of the preposition role classifier that has been trained on the SemEval dataset with and without joint constraints. Note that Table 2 in Section 3, shows the drop in performance when applying the preposition sense classifier. We see that the SemEval-trained preposition role classifier (baseline in the table) achieves an accuracy of 53.29% when tested on the Treebank dataset. Using this classifier jointly with the verb SRL classifier via joint constraints gets an improvement of almost 3 percent in accuracy.

Setting	Preposition Role (Accuracy)
Baseline	53.29
Joint inference	**56.22**

Table 5: Performance of the SemEval-trained preposition role classifier, when tested on the Treebank dataset with and without joint inference with the verb SRL system. The improvement, in this case is statistically significant at the 0.01 level using the sign test.

The primary reason for this improvement, even without re-training the classifier, is that the constraints are defined using only the labels of the systems. This avoids the standard adaptation problems of differing vocabularies and unseen features.

6 Discussion and Related work

Roth and Yih (2004) formulated the problem of extracting entities and relations as an integer linear program, allowing them to use global structural constraints at inference time even though the component classifiers were trained independently. In this paper, we use this idea to combine classifiers that were trained for two different tasks on different datasets using constraints to encode linguistic knowledge.

In the recent years, we have seen several joint models that combine two or more NLP tasks . Andrew et al. (2004) studied verb subcategorization and sense disambiguation of verbs by treating it as a problem of learning with partially labeled structures and proposed to use EM to train the joint model. Finkel and Manning (2009) modeled the task of named entity recognition together with parsing. Meza-Ruiz and Riedel (2009) modeled verb SRL, predicate identification and predicate sense recognition jointly using Markov Logic. Henderson et al. (2008) was designed for jointly learning to predict syntactic and semantic dependencies. Dahlmeier et

al. (2009) addressed the problem of jointly learning verb SRL and preposition sense using the Penn Treebank annotation that was introduced in that work. The key difference between these and the model presented in this paper lies in the simplicity of our model and its easy extensibility because it leverages existing trained systems. Moreover, our model has the advantage that the complexity of the joint parameters is small, hence does not require a large jointly labeled dataset to train the scaling parameters.

Our approach is conceptually similar to that of Rush et al. (2010), which combined separately trained models by enforcing agreement using global inference and solving its linear programming relaxation. They applied this idea to jointly predict dependency and phrase structure parse trees and on the task of predicting full parses together with part-of-speech tags. The main difference in our approach is that we treat the scaling problem as a separate learning problem in itself and train a joint model specifically for re-scaling the output of the trained systems.

The SRL combination system of Surdeanu et al. (2007) studied the combination of three different SRL systems using constraints and also by training secondary scoring functions over the individual systems. Their approach is similar to the one presented in this paper in that, unlike standard reranking, as in Collins (2000), we entertain all possible solutions during inference, while reranking approaches train a discriminative scorer for the top-K solutions of an underlying system. Unlike the SRL combination system, however, our approach spans multiple phenomena. Moreover, in contrast to their re-scoring approaches, we do not define joint features drawn from the predictions of the underlying components to define our global model.

We consider the tasks verb SRL and preposition roles and combine their predictions to provide a richer semantic annotation of text. This approach can be easily extended to include systems that predict structures for other linguistic phenomena because we do not retrain the underlying systems. The semantic relations can be enriched by incorporating more linguistic phenomena such as nominal SRL, defined by the Nombank annotation scheme of Meyers et al. (2004), the preposition function analysis of O'Hara and Wiebe (2009) and noun compound analysis as defined by Girju (2007) and Girju et al. (2009) and others. This presents an exciting direction for future work.

7 Conclusion

This paper presents a strategy for extending semantic role labeling without the need for extensive retraining or data annotation. While standard semantic role labeling focuses on verb and nominal relations, sentences can express relations using other lexical items also. Moreover, the different relations interact with each other and constrain the possible structures that they can take. We use this intuition to define a joint model for inference. We instantiate our model using verb semantic role labeling and preposition role labeling and show that, using linguistic constraints between the tasks and minimal joint learning, we can improve the performance of both tasks. The main advantage of our approach is that we can use existing trained models without re-training them, thus making it easy to extend this work to include other linguistic phenomena.

Acknowledgments

The authors thank the members of the Cognitive Computation Group at the University of Illinois for insightful discussions and the anonymous reviewers for valuable feedback.

This research is supported by the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, ndings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reect the view of the DARPA, AFRL, or the US government.

References

G. Andrew, T. Grenager, and C. D. Manning. 2004. Verb sense and subcategorization: Using joint inference to improve performance on complementary tasks. In *Proceedings of EMNLP*.

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared tasks: Semantic role labeling. In *Proceedings of CoNLL-2004*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*.

M. Chang, L. Ratinov, and D. Roth. 2011. Structured learning with constrained conditional models. *Machine Learning (To appear)*.

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

M. Collins. 2000. Discriminative reranking for natural language parsing. In *ICML*.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

D. Dahlmeier, H. T. Ng, and T. Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In *EMNLP*.

J. R. Finkel and C. D. Manning. 2009. Joint parsing and named entity recognition. In *NAACL*.

R. Girju, P. Nakov, V. Nastase, S. Szpakowicz, P. Turney, and D. Yuret. 2009. Classification of semantic relations between nominals. *Language Resources and Evaluation*.

R. Girju. 2007. Improving the interpretation of noun phrases with cross-linguistic information. In *ACL*.

Y. Goldberg and M. Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *NAACL*.

J. Henderson, P. Merlo, G. Musillo, and I. Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *CoNLL*.

D. Hovy, S. Tratz, and E. Hovy. 2010. What's in a preposition? dimensions of sense disambiguation for an interesting word class. In *Coling 2010: Posters*.

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

K. Litkowski and O. Hargraves. 2005. The preposition project. In *Proceedings of the Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*.

K. Litkowski and O. Hargraves. 2007. Semeval-2007 task 06: Word-sense disambiguation of prepositions. In *SemEval-2007: 4th International Workshop on Semantic Evaluations*.

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

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*.

I Meza-Ruiz and S. Riedel. 2009. Jointly identifying predicates, arguments and senses using markov logic. In *NAACL*.

T. O'Hara and J. Wiebe. 2009. Exploiting semantic role resources for preposition disambiguation. *Computational Linguistics*, 35(2), June.

M. Palmer, P. Kingsbury, and D. Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

P. Pantel and D. Lin. 2002. Discovering word senses from text. In *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *IJCAI*.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP*.

S. Riedel. 2009. Cutting plane map inference for markov logic. In *SRL 2009*.

N. Rizzolo and D. Roth. 2010. Learning based java for rapid development of nlp systems. In *Language Resources and Evaluation*.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *CoNLL*.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *ICML*.

A.M. Rush, D. Sontag, M. Collins, and T. Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *EMNLP*. Association for Computational Linguistics.

M. Surdeanu, L. Màrquez, X. Carreras, and P. R. Comas. 2007. Combination strategies for semantic role labeling. *J. Artif. Int. Res.*, 29:105–151, June.

K. Toutanova, A. Haghighi, and C. D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2).

S. Tratz and D. Hovy. 2009. Disambiguation of preposition sense using linguistically motivated features. In *NAACL: Student Research Workshop and Doctoral Consortium*.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *EMNLP*.

P. Ye and T. Baldwin. 2007. MELB-YB: Preposition Sense Disambiguation Using Rich Semantic Features. In *SemEval-2007*.

Domain-Assisted Product Aspect Hierarchy Generation: Towards Hierarchical Organization of Unstructured Consumer Reviews

Jianxing Yu[1], **Zheng-Jun Zha**[1], **Meng Wang**[1], **Kai Wang**[2], **Tat-Seng Chua**[1]
[1]School of Computing, National University of Singapore
[2]Institute for Infocomm Research, Singapore
{jianxing, zhazj, wangm, chuats}@comp.nus.edu.sg kwang@i2r.a-star.edu.sg

Abstract

This paper presents a domain-assisted approach to organize various aspects of a product into a hierarchy by integrating domain knowledge (e.g., the product specifications), as well as consumer reviews. Based on the derived hierarchy, we generate a hierarchical organization of consumer reviews on various product aspects and aggregate consumer opinions on these aspects. With such organization, user can easily grasp the overview of consumer reviews. Furthermore, we apply the hierarchy to the task of implicit aspect identification which aims to infer implicit aspects of the reviews that do not explicitly express those aspects but actually comment on them. The experimental results on 11 popular products in four domains demonstrate the effectiveness of our approach.

1 Introduction

With the rapidly expanding e-commerce, most retail Web sites encourage consumers to write reviews to express their opinions on various aspects of products. Huge collections of consumer reviews are now available on the Web. These reviews have become an important resource for both consumers and firms. Consumers commonly seek quality information from online consumer reviews prior to purchasing a product, while many firms use online reviews as an important resource in their product development, marketing, and consumer relationship management. However, the reviews are disorganized, leading to the difficulty in information navigation and knowledge acquisition. It is impractical for user to grasp the overview of consumer reviews and opinions on various aspects of a product from such enormous reviews. Among hundreds of product aspects, it is also inefficient for user to browse consumer reviews and opinions on a specific aspect. Thus, there is a compelling need to organize consumer reviews, so as to transform the reviews into a useful knowledge structure. Since the hierarchy can improve information representation and accessibility (Cimiano, 2006), we propose to organize the aspects of a product into a hierarchy and generate a hierarchical organization of consumer reviews accordingly.

Towards automatically deriving an aspect hierarchy from the reviews, we could refer to traditional hierarchy generation methods in ontology learning, which first identify concepts from the text, then determine the parent-child relations between these concepts using either pattern-based or clustering-based methods (Murthy et al., 2010). However, pattern-based methods usually suffer from inconsistency of parent-child relationships among the concepts, while clustering-based methods often result in low accuracy. Thus, by directly utilizing these methods to generate an aspect hierarchy from consumer reviews, the resulting hierarchy is usually inaccurate, leading to unsatisfactory review organization. On the other hand, domain knowledge of products is now available on the Web. For example, there are more than 248,474 product specifications in the product selling Web site CNet.com (Beckham, 2005). These product specifications cover some product aspects and provide coarse-grained parent-child relations among these aspects. Such domain knowledge is useful to help organize the product as-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 140–150,
Edinburgh, Scotland, UK, July 27–31, 2011.

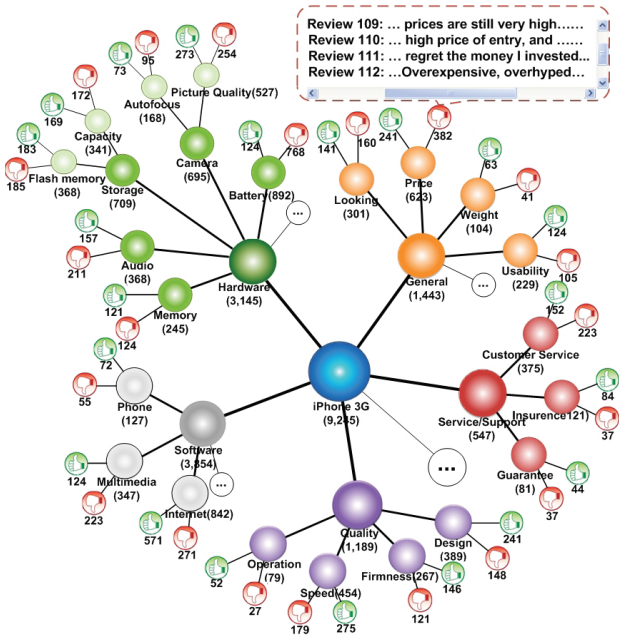


Figure 1: Sample hierarchical organization for iPhone 3G

pects into a hierarchy. However, the initial hierarchy obtained from domain knowledge usually cannot fit the review data well. For example, the initial hierarchy is usually too coarse and may not cover the specific aspects commented in the reviews, while some aspects in the hierarchy may not be of interests to users in the reviews.

Motivated by the above observations, we propose in this paper to organize the product aspects into a hierarchy by simultaneously exploiting the domain knowledge (e.g., the product specification) and consumer reviews. With derived aspect hierarchy, we generate a hierarchical organization of consumer reviews on various aspects and aggregate consumer opinions on these aspects. Figure 1 illustrates a sample of hierarchical review organization for the product "*iPhone 3G*". With such organization, users can easily grasp the overview of product aspects as well as conveniently navigate the consumer reviews and opinions on any aspect. For example, users can find that 623 reviews, out of 9,245 reviews, are about the aspect "*price*", with 241 positive and 382 negative reviews.

Given a collection of consumer reviews on a specific product, we first automatically acquire an initial aspect hierarchy from domain knowledge and identify the aspects from the reviews. Based on the initial hierarchy, we develop a multi-criteria optimization approach to construct an aspect hierarchy to contain all the identified aspects. Our approach incrementally inserts the aspects into the initial hierarchy based on inter-aspect semantic distance, a metric used to measure the semantic relation among aspects. In order to derive reliable semantic distance, we propose to leverage external hierarchies, sampled from WordNet and Open Directory Project, to assist semantic distance learning. With resultant aspect hierarchy, the consumer reviews are then organized to their corresponding aspect nodes in the hierarchy. We then perform sentiment classification to determine consumer opinions on these aspects. Furthermore, we apply the hierarchy to the task of implicit aspect identification. This task aims to infer implicit aspects of the reviews that do not explicitly express those aspects but actually comment on them. For example, the implicit aspect of the review "It is so expensive" is "price." Most existing aspect identification approaches rely on the appearance of aspect terms, and thus are not able to handle implicit aspect problem. Based on our aspect hierarchy, we can infer the implicit aspects by clustering the reviews into their corresponding aspect nodes in the hierarchy. We conduct experiments on 11 popular products in four domains. More details of the corpus are discussed in Section 4. The experimental results demonstrate the effectiveness of our approach.

The main contributions of this work can be summarized as follows:

1) We propose to hierarchically organize consumer reviews according to an aspect hierarchy, so as to transfer the reviews into a useful knowledge structure.

2) We develop a domain-assisted approach to generate an aspect hierarchy by integrating domain knowledge and consumer reviews. In order to derive reliable semantic distance between aspects, we propose to leverage external hierarchies to assist semantic distance learning.

3) We apply the aspect hierarchy to the task of implicit aspect identification, and achieve satisfactory performance.

The rest of this paper is organized as follows. Our approach is elaborated in Section 2 and applied to implicit aspect identification in Section 3. Section 4 presents the evaluations, while Section 5 reviews

related work. Finally, Section 6 concludes this paper with future works.

2 Approach

Our approach consists of four components, including initial hierarchy acquisition, aspect identification, semantic distance learning, and aspect hierarchy generation. Next, we first define some preliminary and notations and then elaborate these components.

2.1 Preliminary and Notations

Preliminary 1. An aspect hierarchy is defined as a tree that consists of a set of unique aspects $\mathcal{A}$ and a set of parent-child relations $\mathcal{R}$ between these aspects.

Given the consumer reviews of a product, let $\mathcal{A} = \{a_1, \cdots, a_k\}$ denotes the product aspects commented in the reviews. $\mathcal{H}^0(\mathcal{A}^0, \mathcal{R}^0)$ denotes the initial hierarchy derived from domain knowledge. It contains a set of aspects $\mathcal{A}^0$ and relations $\mathcal{R}^0$. Our task is to construct an aspect hierarchy $\mathcal{H}(\mathcal{A}, \mathcal{R})$, to cover all the aspects in $\mathcal{A}$ and their parent-child relations $\mathcal{R}$, so that the consumer reviews are hierarchically organized. Note that $\mathcal{H}^0$ can be empty.

2.2 Initial Hierarchy Acquisition

As aforementioned, product specifications on product selling websites cover some product aspects and coarse-grained parent-child relations among these aspects. Such domain knowledge is useful to help organize aspects into a hierarchy. We here employ the approach proposed by Ye and Chua (2006) to automatically acquire an initial aspect hierarchy from the product specifications. The method first identifies the Web page region covering product descriptions and removes the irrelevant contents from the Web page. It then parses the region containing the product information to identify the aspects as well as their structure. Based on the aspects and their structure, it generates an aspect hierarchy.

2.3 Aspect Identification

To identify aspects in consumer reviews, we first parse each review using the Stanford parser [1]. Since the aspects in consumer reviews are usually noun or noun phrases (Liu, 2009), we extract the noun phrases (*NP*) from the parse tree as aspect candidates. While these candidates may contain much noise, we leverage *Pros* and *Cons* reviews (see Figure 2), which are prevalent in forum Web sites, to assist identify aspects from the candidates. It has been shown that simply extracting the frequent noun terms from the *Pros* and *Cons* reviews can get high accurate aspect terms (Liu el al., 2005). Thus, we extract the frequent noun terms from *Pros* and *Cons* reviews as features, then train a one-class *SVM* (Manevitz et al., 2002) to identify aspects from the candidates. While the obtained aspects may contain some synonym terms, such as "earphone" and "headphone", we further perform synonym clustering to get unique aspects. Specifically, we first expand each aspect term with its synonym terms obtained from the synonym terms Web site [2], then cluster them to obtain unique aspects based on unigram feature.

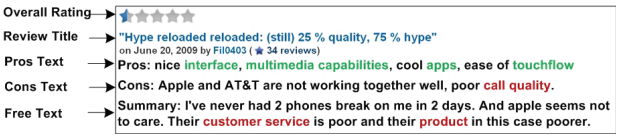


Figure 2: Sample Pros and Cons reviews

2.4 Semantic Distance Learning

Our aspect hierarchy generation approach is essentially based on the semantic relations among aspects. We here define a metric, *Semantic Distance*, $d(a_x, a_y)$, to quantitatively measure the semantic relation between aspects a_x and a_y. We formulate $d(a_x, a_y)$ as the weighted sum of some underlying features,

$$d(a_x, a_y) = \sum\nolimits_j w_j f_j(a_x, a_y), \qquad (1)$$

where w_j is the weight for j-th feature function $f_j(\cdot)$.

Next, we first introduce the linguistic features used in our work and then present the semantic distance learning algorithm that aims to find the optimal weights in Eq.(1).

[1] http://nlp.stanford.edu/software/lex-parser.shtml

[2] http://thesaurus.com

2.4.1 Linguistic Features

Given two aspects a_x and a_y, a feature is defined as a function generating a numeric score $f(a_x, a_y)$ or a vector of scores. The features include *Contextual*, *Co-occurrence*, *Syntactic*, *Pattern* and *Lexical* features (Yang and Callan, 2009). These features are generated based on auxiliary documents collected from Web.

Specifically, we issue each aspect term and aspect term pair as queries to Google and Wikipedia, respectively, and collect the top 100 returned documents of each query. We then split the documents into sentences. Based on these documents and sentences, the features are generated as follows.

Contextual features. For each aspect, we collect the documents containing the aspect as context to build a unigram language model without smoothing. Given two aspects, the KL-divergence between their language models is computed as the *Global-Context* feature between them. Similarly, we collect the left two and right two words surrounding each aspect as context and build a unigram language model. The KL-divergence between the language models of two aspects is defined as the *Local-Context* feature.

Co-occurrence features. We measure the co-occurrence of two aspects by Pointwise Mutual Information (PMI): *PMI*(a_x,a_y)=log(*Count*(a_x,a_y)/ *Count*(a_x) *Count*(a_y)), where *Count*($\cdot$) stands for the number of documents or sentences containing the aspect(s), or the number of Google document hits for the aspect(s). Based on different definitions of *Count*($\cdot$), we define the features of *Document PMI*, *Sentence PMI*, and *Google PMI*, respectively.

Syntactic features. We parse the sentences that contain each aspect pair into syntactic trees via the Stanford Parser. The *Syntactic-path* feature is defined as the average length of the shortest syntactic path between the aspect pair in the tree. In addition, for each aspect, we collect a set of sentences containing it, and label the semantic role of the sentences via ASSERT parser [3]. Given two aspects, the number of the Subject terms overlaps between their sentence sets is computed as the *Subject Overlap* feature. Similarly, for other semantic roles, such as objects, modifiers, and verbs, we define the features of *Object Overlap*, *Modifier Overlap*, and *Verb Overlap*, respectively.

Pattern features. 46 patterns are used in our work, including 6 patterns indicating the hypernym relations of two aspects (Hearst, 1992), and 40 patterns measuring the part-of relations of two aspects (Girju et al., 2006). These pattern features are asymmetric, and they take the parent-child relations among the aspects into consideration. All the patterns are listed in Appendix A (submitted as supplementary material). Based on these patterns, a 46-dimensional score vector is obtained for each aspect pair. A score is 1 if two aspects match a pattern, and 0 otherwise.

Lexical features. We take the word length difference between two aspects, as *Length Difference* feature. In addition, we issue the query "define:aspect" to Google, and collect the definition of each aspect. We then count the word overlaps between the definitions of two aspects, as *Definition Overlap* feature.

2.4.2 Semantic Distance Learning

This section elaborates the learning algorithm that optimizes the semantic distance metric, i.e., the weighting parameters in Eq.(1). Typically, we can utilize the initial hierarchy as training data. The ground-truth distance between two aspects $d^{\mathcal{G}}(a_x, a_y)$ is generated by summing up all the edge distances along the shortest path between a_x and a_y, where every edge weight is assumed as 1. The distance metric is then obtained by solving the following optimization problem,

$$\arg\min_{w_j|_{j=1}^m} \sum_{\substack{a_x,a_y\in\mathcal{A}^0 \\ x<y}} (d^{\mathcal{G}}(a_x,a_y) - \sum_{j=1}^{m} w_j f_j(a_x,a_y))^2 + \eta \cdot \sum_{j=1}^{m} w_j^2, \tag{2}$$

where m is the dimension of linguistic feature, η is a tradeoff parameter. Eq.(2) can be rewrote to its matrix form as,

$$\arg\min_{\mathbf{w}} \left\|\mathbf{d} - \mathbf{f}^T\mathbf{w}\right\|^2 + \eta \cdot \|\mathbf{w}\|^2, \tag{3}$$

where vector $\mathbf{d}$ contains the ground-truth distance of all the aspect pairs. Each element corresponds to the distance of certain aspect pair, and $\mathbf{f}$ is the corresponding feature vector. The optimal solution of $\mathbf{w}$ is given as

$$\mathbf{w}^* = (\mathbf{f}^T\mathbf{f} + \eta \cdot \mathbf{I})^{-1}(\mathbf{f}^T\mathbf{d}) \tag{4}$$

[3] http://cemantix.org/assert.html

where $\mathbf{I}$ is the identity metric.

The above learning algorithm can perform well when sufficient training data (i.e., aspect (term) pairs) is available. However, the initial hierarchy is usually too coarse and thus cannot provide sufficient information. On the other hand, abundant handcrafted hierarchies are available on the Web, such as WordNet and Open Directory Project (ODP). We here propose to leverage these external hierarchies to assist semantic distance learning. A distance metric $\mathbf{w}_0$ is learned from the external hierarchies using the above algorithm. Since $\mathbf{w}_0$ might be biased to the characteristics of the external hierarchies, directly using $\mathbf{w}_0$ in our task may not perform well. Alternatively, we use $\mathbf{w}_0$ as prior knowledge to assist learning the optimal distance metric $\mathbf{w}$ from the initial hierarchy. The learning problem is formulated as follows,

$$\arg\min_{\mathbf{w}} \left\|\mathbf{d} - \mathbf{f}^T\mathbf{w}\right\|^2 + \eta \cdot \|\mathbf{w}\|^2 + \gamma \cdot \|\mathbf{w} - \mathbf{w}_0\|^2, \tag{5}$$

where η and γ are tradeoff parameters.

The optimal solution of $\mathbf{w}$ can be obtained as

$$\mathbf{w}^* = (\mathbf{f}^T\mathbf{f} + (\eta+\gamma)\cdot\mathbf{I})^{-1}(\mathbf{f}^T\mathbf{d} + \gamma\cdot\mathbf{w}_0). \tag{6}$$

As a result, we can compute the semantic distance between each two aspects according to Eq.(1).

2.5 Aspect Hierarchy Generation

Given the aspects $\mathcal{A} = \{a_1, \cdots, a_k\}$ identified from reviews and the initial hierarchy $\mathcal{H}^0(\mathcal{A}^0, \mathcal{R}^0)$ obtained from domain knowledge, our task is to construct an aspect hierarchy to contain all the aspects in $\mathcal{A}$. Inspired by Yang and Callan (2009), we adopt a multi-criteria optimization approach to incrementally insert the aspects into appropriate positions in the hierarchy based on multiple criteria.

Before going to the details, we first introduce an information function to measure the amount of information carried in a hierarchy. An information function $Info(\mathcal{H})$ is defined as the sum of the semantic distances of all the aspect pairs in the hierarchy (Yang and Callan, 2009).

$$Info(\mathcal{H}(\mathcal{A},\mathcal{R})) = \sum_{x<y;a_x,a_y\in\mathcal{A}} d(a_x,a_y). \tag{7}$$

Based on this information function, we then introduce the following three criteria for aspect insertion: *minimum Hierarchy Evolution*, *minimum Hierarchy Discrepancy* and *minimum Semantic Inconsistency*.

Hierarchy Evolution is designed to monitor the structure evolution of a hierarchy. The hierarchy is incrementally hosting more aspects until all the aspects are allocated. The insertion of a new aspect a into different positions in the current hierarchy $\mathcal{H}^{(i)}$ leads to different new hierarchies. Among these new hierarchies, we here assume that the optimal one $\mathcal{H}^{(i+1)}$ should introduce the least changes of information to $\mathcal{H}^{(i)}$.

$$\hat{\mathcal{H}}^{(i+1)} = \arg\min_{\mathcal{H}^{(i+1)}} \Delta Info(\mathcal{H}^{(i+1)} - \mathcal{H}^{(i)}). \tag{8}$$

By plugging in Eq.(7) and using *least square* to measure the information changes, we have,

$$obj_1 = \arg\min_{\mathcal{H}^{(i+1)}} (\textstyle\sum_{x<y;a_x,a_y\in\mathcal{A}_i\cup\{a\}} d(a_x,a_y) - \sum_{x<y;a_x,a_y\in\mathcal{A}_i} d(a_x,a_y))^2, \tag{9}$$

Hierarchy Discrepancy is used to measure the global changes of the structure. We assume a good hierarchy should bring the least changes to the initial hierarchy,

$$\hat{\mathcal{H}}^{(i+1)} = \arg\min_{\mathcal{H}^{(i+1)}} \frac{\Delta Info(\mathcal{H}^{(i+1)} - \mathcal{H}^{(0)})}{i+1}. \tag{10}$$

We then get,

$$obj_2 = \arg\min_{\mathcal{H}^{(i+1)}} \tfrac{1}{i+1}(\textstyle\sum_{x<y;a_x,a_y\in\mathcal{A}_i\cup\{a\}} d(a_x,a_y) - \sum_{x<y;a_x,a_y\in\mathcal{A}_0} d(a_x,a_y))^2. \tag{11}$$

Semantic Inconsistency is introduced to quantify the inconsistency between the semantic distance estimated via the hierarchy and that computed from the feature functions. We assume that a good hierarchy should precisely reflect the semantic distance between aspects. For two aspects, their semantic distance reflected by the hierarchy is computed as the sum of adjacent distances along the shortest path between them,

$$d^{\mathcal{H}}(a_x,a_y) = \sum\nolimits_{p<q;(a_p,a_q)\in SP(a_x,a_y)} d(a_p,a_q), \tag{12}$$

where $SP(a_x,a_y)$ is the shortest path between the aspects (a_x,a_y), (a_p,a_q) are the adjacent nodes along the path.

We then define the following criteria to find the hierarchy with minimum semantic inconsistency,

$$obj_3 = \arg\min_{\mathcal{H}^{(i+1)}} \sum_{x<y; a_x, a_y \in \mathcal{A}_i \cup \{a\};} (d^{\mathcal{H}}(a_x, a_y) - d(a_x, a_y))^2, \tag{13}$$

where $d(a_x, a_y)$ is the distance computed based on the feature functions in Section 2.4.

Through integrating the above criteria, the multi-criteria optimization framework is formulated as,

$$obj = \arg\min_{\mathcal{H}^{(i+1)}} (\lambda_1 \cdot obj_1 + \lambda_2 \cdot obj_2 + \lambda_3 \cdot obj_3)$$
$$\lambda_1 + \lambda_2 + \lambda_3 = 1; \;\; 0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1. \tag{14}$$

where λ_1, λ_2, λ_3 are the tradeoff parameters.

To summarize, our aspect hierarchy generation process starts from an initial hierarchy and inserts the aspects into it one-by-one until all the aspects are allocated. Each aspect is inserted to the optimal position found by Eq.(14). It is worth noting that the insertion order may influence the result. To avoid such influence, we select the aspect with the least objective function value in Eq.(14) to insert. Based on resultant hierarchy, the consumer reviews are then organized to their corresponding aspect nodes in the hierarchy. We further prune out the nodes without reviews from the hierarchy.

Moreover, we perform sentiment classification to determine consumer opinions on various aspects. In particular, we train a *SVM* sentiment classifier based on the *Pros* and *Cons* reviews described in Section 2.3. We collect sentiment terms in the reviews as features and represent reviews as feature vectors using Boolean weighting. Note that we define sentiment terms as those appear in the sentiment lexicon provided by *MPQA* project (Wilson et al., 2005).

3 Implicit Aspect Identification

In this section, we apply the aspect hierarchy to the task of implicit aspect identification. This task aims to infer the aspects of reviews that do not explicitly express those aspects but actually comment on them (Liu et al. 2005). Take the review "The phone is too large" as an example, the task is to infer its implicit aspect "size." It has been observed that the reviews commenting on a same aspect usually use some same sentiment terms (Su et al., 2008). Therefore, sentiment term is an effective feature for identifying implicit aspects. We here collect the sentiment terms as features to represent each review into a feature vector. For each aspect node in the hierarchy, we define its centroid as the average of its feature vectors, i.e., the feature vectors of all the reviews that are allocated at this node. We then calculate the cosine similarity of each implicit-aspect review to the centriods of all the aspect nodes, and allocate the review into the node with maximum similarity. As a result, the implicit aspect reviews are grouped to their related aspect nodes. In other word, their aspects are identified as the corresponding aspect nodes.

4 Evaluations

In this section, we evaluate the effectiveness of our approach on aspect identification, aspect hierarchy generation, and implicit aspect identification.

4.1 Data and Experimental Setting

The details of our product review corpus are given in Table 1. This corpus contains consumer reviews on 11 popular products in four domains. These reviews were crawled from several prevalent forum Web sites, including cnet.com, viewpoints.com, reevoo.com and gsmarena.com. All of the reviews were posted between June, 2009 and Sep 2010. The aspects of the reviews, as well as the opinions on the aspects were manually annotated. We also invited five annotators to construct the gold-standard hierarchies for the products by providing them the initial hierarchies and the aspects in reviews. The conflicts between annotators were resolved through their discussions. For semantic distance learning, we collected 50 hierarchies from WordNet and ODP, respectively. The details are shown in Table 2. We listed the topics of the hierarchies in Appendix B (submitted as supplementary material).

Product Name	*Domain*	*Review#*	*Sentence#*
Canon EOS 450D (Canon EOS)	camera	440	628
Fujifilm Finepix AX245W (Fujifilm)	camera	541	839
Panasonic Lumix DMC-TZ7 (Panasonic)	camera	650	1,546
Apple MacBook Pro (MacBook)	laptop	552	4,221
Samsung NC10 (Samsung)	laptop	2,712	4,946
Apple iPod Touch 2nd (iPod Touch)	MP3	4,567	10,846
Sony NWZ-S639 16GB (Sony NWZ)	MP3	341	773
BlackBerry Bold 9700 (BlackBerry)	phone	4,070	11,008
iPhone 3GS 16GB (iPhone 3GS)	phone	12,418	43,527
Nokia 5800 XpressMusic (Nokia 5800)	phone	28,129	75,001
Nokia N95	phone	15,939	44,379

Table 1: Statistics of the reviews corpus, # denotes the size of the reviews/sentences

Statistic	*WordNet*	*ODP*
Total # hierarchies	50	50
Total # terms	1,964	2,210
Average # depth	5.5	5.9
Total # related topics	12	16

Table 2: Statistics of the External Hierarchies

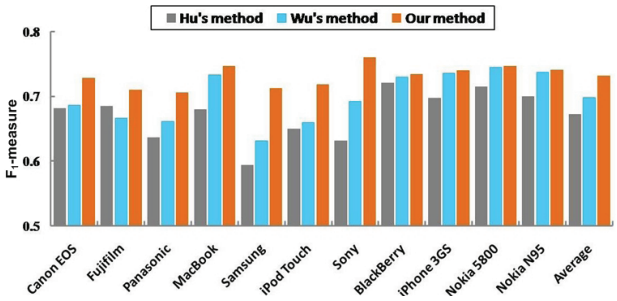


Figure 3: Evaluations on Aspect Identification. t-test, p-values$<$0.05

We employed F_1-measure, which is the combination of *precision* and *recall*, as the evaluation metric for all the evaluations. For the evaluation on aspect hierarchy, we defined *precision* as the percentage of correctly returned parent-child pairs out of the total returned pairs, and *recall* as the percentage of correctly returned parent-child pairs out of the total pairs in the gold standard. Throughout the experiments, we empirically set $\lambda_1 = 0.4$, $\lambda_2 = 0.3$, $\lambda_3 = 0.3$, $\eta = 0.4$ and $\gamma = 0.6$.

4.2 Evaluations on Aspect Identification

We compared our approach against two state-of-the-art methods: a) the method proposed by Hu and Liu (2004), which is based on the association rule mining, and b) the method proposed by Wu et al. (2009), which is based on the dependency parser. The results are presented in Figure 3. On average, our approach significantly outperforms Hu's and Wu's method in terms of F_1-measure by over 5.87% and 3.27%, respectively.

4.3 Evaluations on Aspect Hierarchy

4.3.1 Comparisons with the State-of-the-Arts

We compared our approach against four traditional hierarchy generation methods in the researches on ontology learning, including a) pattern-based method (Hearst, 1992) and b) clustering-based method by Shi et al. (2008), c) the method proposed by Snow et al. (2006) which was based on a probabilistic model, and d) the method proposed by Yang and Callan (2009). Since our approach and Yang's method can utilize initial hierarchy to assist hierarchy generation, we evaluated their performance with or without initial hierarchy, respectively. For the sake of fair comparison, Snow's, Yang's and our methods used the same linguistic features in Section 2.4.1.

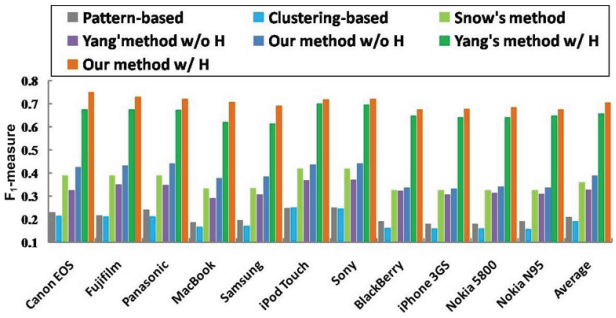


Figure 4: Evaluations on Aspect Hierarchy Generation. t-test, p-values$<$0.05. w/ H denotes the methods with initial hierarchy, accordingly, w/o H refers to the methods without initial hierarchy.

Figure 4 shows the performance comparisons of these five methods. We can see that our approach without using initial hierarchy outperforms the pattern-based, clustering-based, Snow's, and Yang's methods by over 17.9%, 19.8%, 2.9% and 6.1% respectively in terms of average F_1-measure. By exploiting initial hierarchy, our approach improves the performance significantly. As compared to the pattern-based, clustering-based and Snow's methods, it improves the average performance by over 49.4%, 51.2% and 34.3% respectively. Compared to Yang's method with initial hierarchy, it achieves 4.7% improvements on the average performance.

The results show that pattern-based and clustering-based methods perform poor. Pattern-based method may suffer from the problem of low coverage of patterns, especially when the patterns are manually pre-defined, while the clustering-based method (Shi et al., 2008) may sustain to the bisection clustering mechanism which can only generate a binary-tree. The results also illustrate that our approach outperforms Snow's and Yang's methods. By exploiting external hierarchies, our

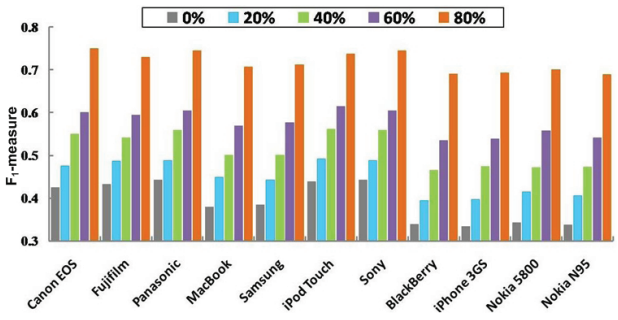


Figure 5: Evaluations on the Impact of Initial Hierarchy. t-test, p-values$<$0.05.

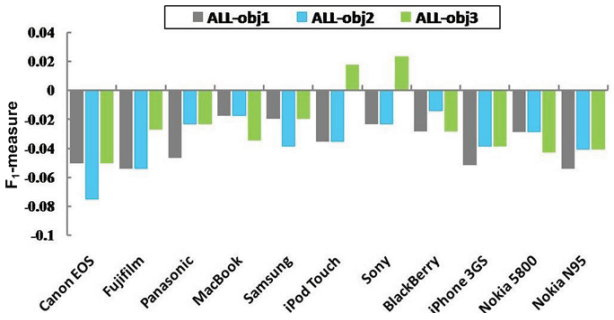


Figure 6: Evaluations of the Optimization Criteria. % of change in F_1-measure when a single criterion is removed. t-test, p-values$<$0.05.

approach is able to derive reliable semantic distance between aspects and thus improve the performance.

4.3.2 Evaluations on Effectiveness of Initial Hierarchy

In this section, we show that even based on a small part of the initial hierarchy, our approach can still generate a satisfactory hierarchy. We explored different proportion of initial hierarchy, including 0%, 20%, 40%, 60% and 80% of the aspect pairs which are collected top-down from the initial hierarchy. As shown in Figure 5, the performance increases when larger proportion of the initial hierarchy is used. Thus, we can speculate that domain knowledge is valuable in aspect hierarchy generation.

4.3.3 Evaluations on Effectiveness of Optimization Criteria

We conducted a leave-one-out study to evaluate the effectiveness of each optimization criterion. In particular, we set one of the tradeoff parameters (λ_1, λ_2, λ_3) in Eq.(14) to zero, and distributed its weight to the rest parameters averagely. From Figure 6, we find that removing any optimization criterion would degrade the performance on most products. It is interesting to note that removing the third optimization criterion, i.e., minimum semantic inconsistency, slightly increases the performance on two products (ipad touch and sony MP3). The reason might be that the values of the three tradeoff parameters (empirically set in Section 4.1) are not suitable for these two products.

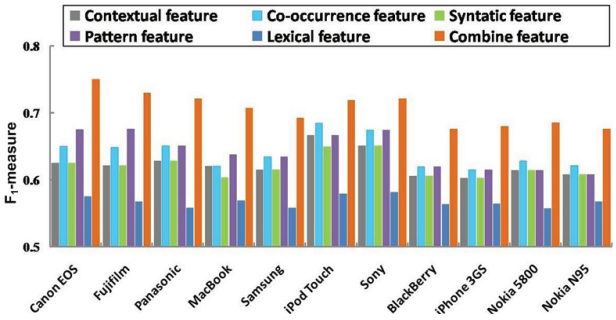


Figure 7: Evaluations on the Impact of Linguistic Features. t-test, p-values$<$0.05.

4.3.4 Evaluations on Semantic Distance Learning

In this section, we evaluated the impact of the features and external hierarchies in semantic distance learning. We investigated five sets of features as described in Section 2.4.1, including contextual, co-occurrence, syntactic, pattern and lexical features. From Figure 7, we observe that the co-occurrence and pattern features perform much better than contextual and syntactic features. A possible reason is that co-occurrence and pattern features are more likely to indicate parent-child aspect relationships, while contextual and syntactic features are probable to measure sibling aspect relationships. Among these features, the lexical features perform the worst. The combination of all the features achieves the best performance.

Next, we evaluated the effectiveness of external hierarchies in semantic distance learning. We compared the performance of our approach with or without the external hierarchies. From Figure 8, we find that by exploiting the external hierarchies, our ap-

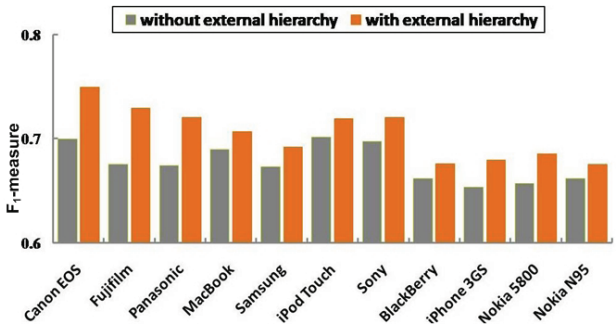


Figure 8: Evaluations on the Impact of External Hierarchy. t-test, p-values$<$0.05.

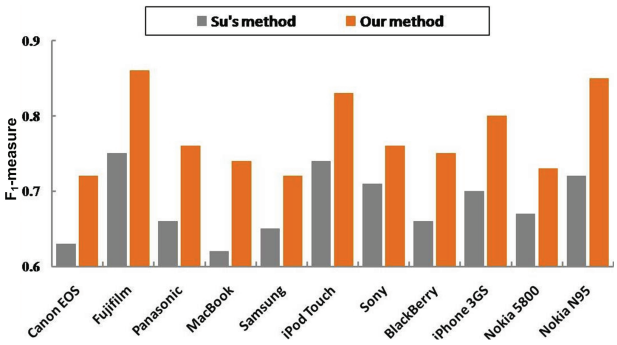


Figure 9: Evaluations on Implicit Aspects Identification. t-test, p-values$<$0.05

proach improves the performance significantly. The improvement is over 2.81% in terms of average F_1-measure. This implies that by using external hierarchies, our approach can obtain effective semantic distance, and thus improve the performance of aspect hierarchy generation.

Additionally, for sentiment classification, our *SVM* classifier achieves an average F_1-measure of 0.787 in the 11 products.

4.4 Evaluations on Implicit Aspect Identification

To evaluate the performance of our approach on implicit aspect identification, we collected 29,657 implicit aspect review sentences on the 11 products from the four forum Web sites introduced in Section 4.1. While most existing approaches for implicit aspect identification rely on hand-crafted rules (Liu, 2009), the method proposed in Su et al. (2008) can identify implicit aspects without hand-crafted rules based on mutual clustering. Therefore, we adopt Su's method as the baseline here. Figure 9 illustrates the performance comparison between Su's and our approach. We can see that our approach outperforms Su's method by over 9.18% in terms of average F_1-measure. This shows that our approach can identify the implicit aspects accurately by exploiting the underlying associations among the sentiment terms and each aspect in the hierarchy.

5 Related Work

Some researches treated review organization as a multi-document summarization problem, and generated a summary by selecting and ordering sentences taken from multiple reviews (Nishikawa et al., 2010). These works did not drill down to the fine-grained level to explore the opinions on the product aspects. Other researchers proposed to produce a summary covering consumer opinions on each aspect. For example, Hu and Liu (2004) focused on extracting the aspects and determining opinions on the aspects. However, their generated summary was unstructured, where the possible relationships between aspects were not recognized (Cadilhac et al., 2010). Subsequently, Carenini et al. (2006) proposed to map the aspect to a user-defined taxonomy, but the taxonomy was hand-crafted which was not scalable.

Different from the previous works, we focus on automatically generating an aspect hierarchy to hierarchically organize consumer reviews. There are some related works on ontology learning, which first identify concepts from text, and then determine parent-child relations between these concepts using either pattern-based or clustering-based methods (Murthy et al., 2010). Pattern-based methods usually defined some lexical syntactic patterns to extract the relations, while clustering-based methods mostly utilized the hierarchical clustering methods to build a hierarchy (Roy et al., 2006). Some works proposed to integrate the pattern-based and clustering-based methods in a general model, such as the probabilistic model (Snow et al., 2006) and metric-based model (Yang and Callan, 2009).

The researches on aspect identification are also related to our work. Various aspect identification methods have been proposed (Popescu et al., 2005), including supervised methods (Liu el al., 2005), and unsupervised methods (Mei et al., 2007). Different

features have been investigated for this task. For example, Wu et al. (2009) identified aspects based on the features explored by dependency parser. For implicit aspect identification, some works proposed to define rules for identification (Liu el al., 2005), while others suggested to automatically generate rules via mutual clustering (Su et al., 2008). On the other hand, there are some related works on sentiment classification (Pang and Lee, 2008). These works can be categorized into four granularities: document-level, sentence-level, aspect-level and word-level sentiment classification (Liu, 2009). Existing researches have been studied unsupervised (Kim et al., 2004), supervised (Pang et al., 2002; Pang et al., 2005) and semi-supervised classification approaches (Goldberg et al., 2006; Li et al., 2009) on these four levels.

6 Conclusions and Future Works

In this paper, we have developed a domain-assisted approach to generate product aspect hierarchy by integrating domain knowledge and consumer reviews. Based on the derived hierarchy, we can generate a hierarchical organization of consumer reviews as well as consumer opinions on the aspects. With such organization, user can easily grasp the overview of consumer reviews, as well as seek consumer reviews and opinions on any specific aspect by navigating through the hierarchy. We have further applied the hierarchy to the task of implicit aspect identification. We have conducted evaluations on 11 different products in four domains. The experimental results have demonstrated the effectiveness of our approach. In the future, we will explore other linguistic features to learn the semantic distance between aspects, as well as apply our approach to other applications.

Acknowledgments

This work is supported by NUS-Tsinghua Extreme Search (NExT) project under the grant number: R-252-300-001-490. We give warm thanks to the project and anonymous reviewers for their valuable comments.

References

P. Beineke, T. Hastie, C. Manning, and S. Vaithyanathan. An Exploration of Sentiment Summarization. *AAAI*, 2003.

J. Beckham. The Cnet E-commerce Data set. *Technical Reports*, 2005.

G. Carenini, R. Ng, and E. Zwart. Multi-document Summarization of Evaluative Text. *ACL*, 2006.

A. Cadilhac, F. Benamara, and N. Aussenac-Gilles. Ontolexical Resources for Feature based Opinion Mining: a Case-study. *Ontolex*, 2010.

P. Cimiano, A. Madche, S. Staab, and J. Volker. Ontology Learning. *Handbook on Ontologies, Springer*, 2004.

P. Cimiano, A. Hotho, and S. Staab. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Artificial Intelligence*, 2005.

P. Cimiano. Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. *Springer-Verlag New York, Inc. Secaucus, NJ, USA*, 2006.

S. Dasgupta and V. Ng. Mine the Easy, Classify the Hard: A Semi-supervised Approach to Automatic Sentiment Classification. *ACL*, 2009.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised Named-entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 2005.

A. Esuli and F. Sebastiani. A Publicly Available Lexical Resource for Opinion Mining. *LREC*, 2006.

M. Gamon, A. Aue, S. Corston-Oliver, and E. Ringger. Pulse: Mining Customer Opinions from Free Text. *IDA*, 2005.

R. Girju and A. Badulescu. Automatic Discovery of Part-whole Relations *Computational Linguistics*, 2006.

A. Goldberg and X. Zhu. Seeing Stars When There Aren't Many Stars: Graph-based Semi-supervised Learning for Sentiment Categorization. *ACL*, 2006.

M.A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. *Coling*, 1992.

M. Hu and B. Liu. Mining and Summarizing Customer Reviews. *SIGKDD*, 2004.

X. Hu, N. Sun, C. Zhang, and T.-S. Chua Exploiting Internal and External Semantics for the Clustering of Short Texts Using World Knowledge. *CIKM*, 2009.

S. Kim and E. Hovy. Determining the Sentiment of Opinions. *COLING*, 2004.

A. C. Konig and E. Brill. Reducing the Human Overhead in Text Categorization. *KDD*, 2006.

Z. Kozareva, E. Riloff, and E. Hovy. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. *ACL*, 2008.

T. Li, Y. Zhang, and V. Sindhwani. A Non-negative Matrix Tri-factorization Approach to Sentiment Classification with Lexical Prior Knowledge. *ACL*, 2009.

B. Liu, M. Hu, and J. Cheng. Opinion Observer: Analyzing and Comparing Opinions on the Web. *WWW*, 2005.

B. Liu. Handbook Chapter: Sentiment Analysis and Subjectivity. Handbook of Natural Language Processing. *Marcel Dekker, Inc. New York, NY, USA*, 2009.

L.M. Manevitz and M. Yousef. One-class SVMs for Document Classification. *Machine Learning*, 2002.

Q. Mei, X. Ling, M. Wondra, H. Su, and C.X. Zhai. Topic Sentiment Mixture: Modeling Facets and Opinions in Weblogs. *WWW*, 2007.

X. Meng and H. Wang. Mining User Reviews: from Specification to Summarization. *ACL-IJCNLP*, 2009.

K. Murthy, T.A. Faruquie, L.V. Subramaniam, K.H. Prasad, and M. Mohania. Automatically Generating Term-frequency-induced Taxonomies. *ACL*, 2010.

H. Nishikawa, T. Hasegawa, Y. Matsuo, and G. Kikui. Optimizing Informativeness and Readability for Sentiment Summarization. *ACL*, 2010.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. *EMNLP*, 2002.

B. Pang and L. Lee. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with respect to Rating Scales. *ACL*, 2005.

B. Pang and L. Lee. *Opinion mining and sentiment analysis*. Foundations and Trends in Information Retrieval, 2008.

HH. Pang, J. Shen, and R. Krishnan Privacy-Preserving, Similarity-Based Text Retrieval. *ACM Transactions on Internet Technology*, 2010.

A.M. Popescu and O. Etzioni. Extracting Product Features and Opinions from Reviews. *HLT/EMNLP*, 2005.

H. Poon and P. Domingos. Unsupervised Ontology Induction from Text. *ACL*, 2010.

G. Qiu, B. Liu, J. Bu, and C. Chen. Expanding Domain Sentiment Lexicon through Double Propagation. *IJCAI*, 2009.

S. Roy and L.V. Subramaniam. Automatic Generation of Domain Models for Call Centers from Noisy Transcriptions. *ACL*, 2009.

B. Shi and K. Chang. Generating a Concept Hierarchy for Sentiment Analysis. *SMC*, 2008.

R. Snow and D. Jurafsky. Semantic Taxonomy Induction from Heterogenous Evidence. *ACL*, 2006.

Q. Su, X. Xu, H. Guo, X. Wu, X. Zhang, B. Swen, and Z. Su. Hidden Sentiment Association in Chinese Web Opinion Mining. *WWW*, 2008.

I. Titov and R. McDonald. A Joint Model of Text and Aspect Ratings for Sentiment Summarization. *ACL*, 2008.

P. Turney. Thumbs up or thumbs down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *ACL*, 2002.

Y. Wu, Q. Zhang, X. Huang, and L. Wu. Phrase Dependency Parsing for Opinion Mining. *ACL*, 2009.

T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. *HLT/EMNLP*, 2005.

H. Yang and J. Callan A Metric-based Framework for Automatic Taxonomy Induction. *ACL*, 2009.

S. Ye and T.-S. Chua. Learning Object Models from Semi-structured Web Documents. *IEEE Transactions on Knowledge and Data Engineering*, 2006.

J. Yi, T. Nasukawa, W. Niblack, and R. Bunescu. Sentiment Analyzer: Extracting Sentiments about a Given Topic using Natural Language Processing Techniques. *ICDM*, 2003.

L. Zhuang,F. Jing, and X.Y. Zhu Movie Review Mining and Summarization *CIKM*, 2006.

Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions

Richard Socher **Jeffrey Pennington*** **Eric H. Huang** **Andrew Y. Ng** **Christopher D. Manning**
Computer Science Department, Stanford University, Stanford, CA 94305, USA
*SLAC National Accelerator Laboratory, Stanford University, Stanford, CA 94309, USA
richard@socher.org {jpennin,ehhuang,ang,manning}@stanford.edu ang@cs.stanford.edu

Abstract

We introduce a novel machine learning framework based on recursive autoencoders for sentence-level prediction of sentiment label distributions. Our method learns vector space representations for multi-word phrases. In sentiment prediction tasks these representations outperform other state-of-the-art approaches on commonly used datasets, such as movie reviews, without using any pre-defined sentiment lexica or polarity shifting rules. We also evaluate the model's ability to predict sentiment distributions on a new dataset based on confessions from the experience project. The dataset consists of personal user stories annotated with multiple labels which, when aggregated, form a multinomial distribution that captures emotional reactions. Our algorithm can more accurately predict distributions over such labels compared to several competitive baselines.

1 Introduction

The ability to identify sentiments about personal experiences, products, movies etc. is crucial to understand user generated content in social networks, blogs or product reviews. Detecting sentiment in these data is a challenging task which has recently spawned a lot of interest (Pang and Lee, 2008).

Current baseline methods often use bag-of-words representations which cannot properly capture more complex linguistic phenomena in sentiment analysis (Pang et al., 2002). For instance, while the two phrases "white blood cells destroying an infection" and "an infection destroying white blood cells" have the same bag-of-words representation, the former is a positive reaction while the later is very negative. More advanced methods such as (Nakagawa et al., 2010) that can capture such phenomena use many manually constructed resources (sentiment lexica, parsers, polarity-shifting rules). This limits the applicability of these methods to a broader range of tasks and languages. Lastly, almost all previous work is based on single, positive/negative categories or scales such as star ratings. Examples are movie reviews (Pang and Lee, 2005), opinions (Wiebe et al., 2005), customer reviews (Ding et al., 2008) or multiple aspects of restaurants (Snyder and Barzilay, 2007). Such a one-dimensional scale does not accurately reflect the complexity of human emotions and sentiments.

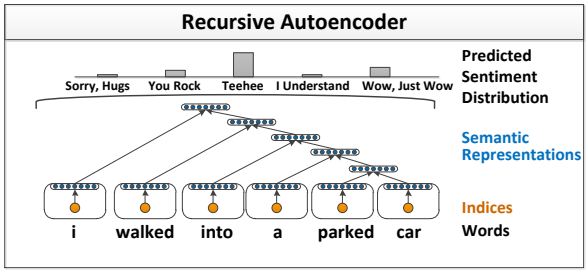


Figure 1: Illustration of our recursive autoencoder architecture which learns semantic vector representations of phrases. Word indices (orange) are first mapped into a semantic vector space (blue). Then they are recursively merged by the same autoencoder network into a fixed length sentence representation. The vectors at each node are used as features to predict a distribution over sentiment labels.

In this work, we seek to address three issues. (i) Instead of using a bag-of-words representation, our model exploits hierarchical structure and uses compositional semantics to understand sentiment. (ii) Our system can be trained both on unlabeled domain data and on supervised sentiment data and does not require any language-specific sentiment lexica,

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 151–161,
Edinburgh, Scotland, UK, July 27–31, 2011.

parsers, etc. (iii) Rather than limiting sentiment to a positive/negative scale, we predict a multidimensional distribution over several complex, interconnected sentiments.

We introduce an approach based on semi-supervised, recursive autoencoders (RAE) which use as input continuous word vectors. Fig. 1 shows an illustration of the model which learns vector representations of phrases and full sentences as well as their hierarchical structure from unsupervised text. We extend our model to also learn a distribution over sentiment labels at each node of the hierarchy.

We evaluate our approach on several standard datasets where we achieve state-of-the art performance. Furthermore, we show results on the recently introduced experience project (EP) dataset (Potts, 2010) that captures a broader spectrum of human sentiments and emotions. The dataset consists of very personal confessions anonymously made by people on the experience project website *www.experienceproject.com*. Confessions are labeled with a set of five reactions by other users. Reaction labels are *you rock* (expressing approvement), *tehee* (amusement), *I understand*, *Sorry, hugs* and *Wow, just wow* (displaying shock). For evaluation on this dataset we predict both the label with the most votes as well as the full distribution over the sentiment categories. On both tasks our model outperforms competitive baselines. A set of over 31,000 confessions as well as the code of our model are available at *www.socher.org*.

After describing the model in detail, we evaluate it qualitatively by analyzing the learned n-gram vector representations and compare quantitatively against other methods on standard datasets and the EP dataset.

2 Semi-Supervised Recursive Autoencoders

Our model aims to find vector representations for variable-sized phrases in either unsupervised or semi-supervised training regimes. These representations can then be used for subsequent tasks. We first describe neural word representations and then proceed to review a related recursive model based on autoencoders, introduce our recursive autoencoder (RAE) and describe how it can be modified to jointly learn phrase representations, phrase structure and sentiment distributions.

2.1 Neural Word Representations

We represent words as continuous vectors of parameters. We explore two settings. In the first setting we simply initialize each word vector $x \in \mathbb{R}^n$ by sampling it from a zero mean Gaussian distribution: $x \sim \mathcal{N}(0, \sigma^2)$. These word vectors are then stacked into a word embedding matrix $L \in \mathbb{R}^{n \times |V|}$, where $|V|$ is the size of the vocabulary. This initialization works well in supervised settings where a network can subsequently modify these vectors to capture certain label distributions.

In the second setting, we pre-train the word vectors with an unsupervised neural language model (Bengio et al., 2003; Collobert and Weston, 2008). These models jointly learn an embedding of words into a vector space and use these vectors to predict how likely a word occurs given its context. After learning via gradient ascent the word vectors capture syntactic and semantic information from their co-occurrence statistics.

In both cases we can use the resulting matrix of word vectors L for subsequent tasks as follows. Assume we are given a sentence as an ordered list of m words. Each word has an associated vocabulary index k into the embedding matrix which we use to retrieve the word's vector representation. Mathematically, this look-up operation can be seen as a simple projection layer where we use a binary vector b which is zero in all positions except at the kth index,

$$x_i = Lb_k \in \mathbb{R}^n. \quad (1)$$

In the remainder of this paper, we represent a sentence (or any n-gram) as an ordered list of these vectors $(x_1, \ldots, x_m)$. This word representation is better suited to autoencoders than the binary number representations used in previous related autoencoder models such as the recursive autoassociative memory (RAAM) model (Pollack, 1990; Voegtlin and Dominey, 2005) or recurrent neural networks (Elman, 1991) since sigmoid units are inherently continuous. Pollack circumvented this problem by having vocabularies with only a handful of words and by manually defining a threshold to binarize the resulting vectors.

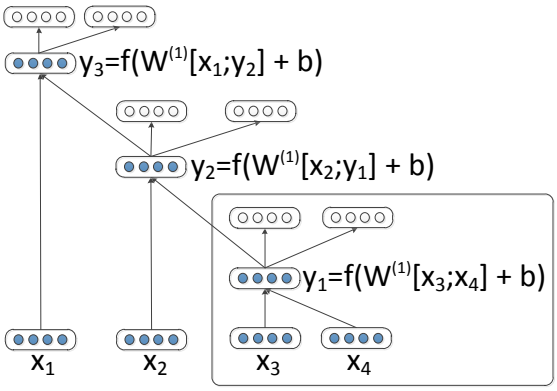


Figure 2: Illustration of an application of a recursive autoencoder to a binary tree. The nodes which are not filled are only used to compute reconstruction errors. A standard autoencoder (in box) is re-used at each node of the tree.

2.2 Traditional Recursive Autoencoders

The goal of autoencoders is to learn a representation of their inputs. In this section we describe how to obtain a reduced dimensional vector representation for sentences.

In the past autoencoders have only been used in setting where the tree structure was given a-priori. We review this setting before continuing with our model which does not require a given tree structure. Fig. 2 shows an instance of a recursive autoencoder (RAE) applied to a given tree. Assume we are given a list of word vectors $x = (x_1, \ldots, x_m)$ as described in the previous section as well as a binary tree structure for this input in the form of branching triplets of parents with children: $(p \rightarrow c_1 c_2)$. Each child can be either an input word vector x_i or a nonterminal node in the tree. For the example in Fig. 2, we have the following triplets: $((y_1 \rightarrow x_3 x_4), (y_2 \rightarrow x_2 y_1), (y_1 \rightarrow x_1 y_2))$. In order to be able to apply the same neural network to each pair of children, the hidden representations y_i have to have the same dimensionality as the x_i's.

Given this tree structure, we can now compute the parent representations. The first parent vector y_1 is computed from the children $(c_1, c_2) = (x_3, x_4)$:

$$p \quad = \quad f(W^{(1)}[c_1; c_2] + b^{(1)}), \tag{2}$$

where we multiplied a matrix of parameters $W^{(1)} \in \mathbb{R}^{n \times 2n}$ by the concatenation of the two children. After adding a bias term we applied an element-wise activation function such as $\tanh$ to the resulting vector. One way of assessing how well this n-dimensional vector represents its children is to try to reconstruct the children in a reconstruction layer:

$$\left[c_1'; c_2'\right] \quad = \quad W^{(2)} p + b^{(2)}. \tag{3}$$

During training, the goal is to minimize the reconstruction errors of this input pair. For each pair, we compute the Euclidean distance between the original input and its reconstruction:

$$E_{rec}([c_1; c_2]) = \frac{1}{2} \left|\left|[c_1; c_2] - \left[c_1'; c_2'\right]\right|\right|^2. \tag{4}$$

This model of a standard autoencoder is boxed in Fig. 2. Now that we have defined how an autoencoder can be used to compute an n-dimensional vector representation (p) of two n-dimensional children (c_1, c_2), we can describe how such a network can be used for the rest of the tree.

Essentially, the same steps repeat. Now that y_1 is given, we can use Eq. 2 to compute y_2 by setting the children to be $(c_1, c_2) = (x_2, y_1)$. Again, after computing the intermediate parent vector y_2, we can assess how well this vector capture the content of the children by computing the reconstruction error as in Eq. 4. The process repeat until the full tree is constructed and we have a reconstruction error at each nonterminal node. This model is similar to the RAAM model (Pollack, 1990) which also requires a fixed tree structure.

2.3 Unsupervised Recursive Autoencoder for Structure Prediction

Now, assume there is no tree structure given for the input vectors in x. The goal of our structure-prediction RAE is to minimize the reconstruction error of all vector pairs of children in a tree. We define $A(x)$ as the set of all possible trees that can be built from an input sentence x. Further, let $T(y)$ be a function that returns the triplets of a tree indexed by s of all the non-terminal nodes in a tree. Using the reconstruction error of Eq. 4, we compute

$$RAE_\theta(x) = \underset{y \in A(x)}{\arg\min} \sum_{s \in T(y)} E_{rec}([c_1; c_2]_s) \tag{5}$$

We now describe a greedy approximation that constructs such a tree.

Greedy Unsupervised RAE. For a sentence with m words, we apply the autoencoder recursively. It takes the first pair of neighboring vectors, defines them as potential children of a phrase $(c_1; c_2) = (x_1; x_2)$, concatenates them and gives them as input to the autoencoder. For each word pair, we save the potential parent node p and the resulting reconstruction error.

After computing the score for the first pair, the network is shifted by one position and takes as input vectors $(c_1, c_2) = (x_2, x_3)$ and again computes a potential parent node and a score. This process repeats until it hits the last pair of words in the sentence: $(c_1, c_2) = (x_{m-1}, x_m)$. Next, it selects the pair which had the lowest reconstruction error (E_{rec}) and its parent representation p will represent this phrase and replace both children in the sentence word list. For instance, consider the sequence (x_1, x_2, x_3, x_4) and assume the lowest E_{rec} was obtained by the pair (x_3, x_4). After the first pass, the new sequence then consists of $(x_1, x_2, p_{(3,4)})$. The process repeats and treats the new vector $p_{(3,4)}$ like any other input vector. For instance, subsequent states could be either: $(x_1, p_{(2,(3,4))})$ or $(p_{(1,2)}, p_{(3,4)})$. Both states would then finish with a deterministic choice of collapsing the remaining two states into one parent to obtain $(p_{(1,(2,(3,4)))})$ or $(p_{((1,2),(3,4))})$ respectively. The tree is then recovered by unfolding the collapsing decisions.

The resulting tree structure captures as much of the single-word information as possible (in order to allow reconstructing the word vectors) but does not necessarily follow standard syntactic constraints. We also experimented with a method that finds better solutions to Eq. 5 based on CKY-like beam search algorithms (Socher et al., 2010; Socher et al., 2011) but the performance is similar and the greedy version is much faster.

Weighted Reconstruction. One problem with simply using the reconstruction error of both children equally as describe in Eq. 4 is that each child could represent a different number of previously collapsed words and is hence of bigger importance for the overall meaning reconstruction of the sentence. For instance in the case of $(x_1, p_{(2,(3,4))})$ one would like to give more importance to reconstructing p than x_1. We capture this desideratum by adjusting the reconstruction error. Let n_1, n_2 be the number of words underneath a current potential child, we re-define the reconstruction error to be $E_{rec}([c_1; c_2]; \theta) =$

$$\frac{n_1}{n_1+n_2}\left|\left|c_1 - c_1'\right|\right|^2 + \frac{n_2}{n_1+n_2}\left|\left|c_2 - c_2'\right|\right|^2 \quad (6)$$

Length Normalization. One of the goals of RAEs is to induce semantic vector representations that allow us to compare n-grams of different lengths. The RAE tries to lower reconstruction error of not only the bigrams but also of nodes higher in the tree. Unfortunately, since the RAE computes the hidden representations it then tries to reconstruct, it can just lower reconstruction error by making the hidden layer very small in magnitude. To prevent such undesirable behavior, we modify the hidden layer such that the resulting parent representation always has length one, after computing p as in Eq. 2, we simply set: $p = \frac{p}{||p||}$.

2.4 Semi-Supervised Recursive Autoencoders

So far, the RAE was completely unsupervised and induced general representations that capture the semantics of multi-word phrases.In this section, we extend RAEs to a semi-supervised setting in order to predict a sentence- or phrase-level target distribution t.[1]

One of the main advantages of the RAE is that each node of the tree built by the RAE has associated with it a distributed vector representation (the parent vector p) which could also be seen as features describing that phrase. We can leverage this representation by adding on top of each parent node a simple $\mathrm{softmax}$ layer to predict class distributions:

$$d(p; \theta) = \mathrm{softmax}(W^{label} p). \quad (7)$$

Assuming there are K labels, $d \in \mathbb{R}^K$ is a K-dimensional multinomial distribution and $\sum_{k=1} d_k = 1$. Fig. 3 shows such a semi-supervised RAE unit. Let t_k be the kth element of the multinomial target label distribution t for one entry. The $\mathrm{softmax}$ layer's outputs are interpreted as conditional probabilities $d_k = p(k|[c_1; c_2])$, hence the cross-entropy error is

$$E_{cE}(p, t; \theta) = -\sum_{k=1}^{K} t_k \log d_k(p; \theta). \quad (8)$$

[1]For the binary label classification case, the distribution is of the form $[1, 0]$ for class 1 and $[0, 1]$ for class 2.

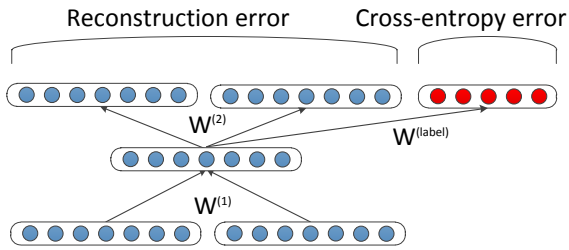


Figure 3: Illustration of an RAE unit at a nonterminal tree node. Red nodes show the supervised *softmax* layer for label distribution prediction.

Using this cross-entropy error for the label and the reconstruction error from Eq. 6, the final semi-supervised RAE objective over (sentences,label) pairs (x, t) in a corpus becomes

$$J = \frac{1}{N}\sum_{(x,t)} E(x,t;\theta) + \frac{\lambda}{2}||\theta||^2, \qquad (9)$$

where we have an error for each entry in the training set that is the sum over the error at the nodes of the tree that is constructed by the greedy RAE:

$$E(x,t;\theta) = \sum_{s \in T(RAE_\theta(x))} E([c_1;c_2]_s, p_s, t, \theta).$$

The error at each nonterminal node is the weighted sum of reconstruction and cross-entropy errors, $E([c_1;c_2]_s, p_s, t, \theta) =$

$$\alpha E_{rec}([c_1;c_2]_s;\theta) + (1-\alpha)E_{cE}(p_s, t;\theta).$$

The hyperparameter α weighs reconstruction and cross-entropy error. When minimizing the cross-entropy error of this softmax layer, the error will backpropagate and influence both the RAE parameters and the word representations. Initially, words such as *good* and *bad* have very similar representations. This is also the case for Brown clusters and other methods that use only cooccurrence statistics in a small window around each word. When learning with positive/negative sentiment, the word embeddings get modified and capture less syntactic and more sentiment information.

In order to predict the sentiment distribution of a sentence with this model, we use the learned vector representation of the top tree node and train a simple logistic regression classifier.

3 Learning

Let $\theta = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(1)}, W^{label}, L)$ be the set of our model parameters, then the gradient becomes:

$$\frac{\partial J}{\partial \theta} = \frac{1}{N}\sum_{(x,t)} \frac{\partial E(x,t;\theta)}{\partial \theta} + \lambda\theta. \qquad (10)$$

To compute this gradient, we first greedily construct all trees and then derivatives for these trees are computed efficiently via backpropagation through structure (Goller and Küchler, 1996). Because the algorithm is greedy and the derivatives of the supervised cross-entropy error also modify the matrix $W^{(1)}$, this objective is not necessarily continuous and a step in the gradient descent direction may not necessarily decrease the objective. However, we found that L-BFGS run over the complete training data (batch mode) to minimize the objective works well in practice, and that convergence is smooth, with the algorithm typically finding a good solution quickly.

4 Experiments

We first describe the new experience project (EP) dataset, results of standard classification tasks on this dataset and how to predict its sentiment label distributions. We then show results on other commonly used datasets and conclude with an analysis of the important parameters of the model.

In all experiments involving our model, we represent words using 100-dimensional word vectors. We explore the two settings mentioned in Sec. 2.1. We compare performance on standard datasets when using randomly initialized word vectors (random word init.) or word vectors trained by the model of Collobert and Weston (2008) and provided by Turian et al. (2010).[2] These vectors were trained on an unlabeled corpus of the English Wikipedia. Note that alternatives such as Brown clusters are not suitable since they do not capture sentiment information (*good* and *bad* are usually in the same cluster) and cannot be modified via backpropagation.

[2] http://metaoptimize.com/projects/wordreprs/

Corpus	K	Instances	Distr.(+/-)	Avg$\|W\|$
MPQA	2	10,624	0.31/0.69	3
MR	2	10,662	0.5/0.5	22
EP	5	31,675	.2/.2/.1/.4/.1	113
EP≥ 4	5	6,129	.2/.2/.1/.4/.1	129

Table 1: Statistics on the different datasets. K is the number of classes. Distr. is the distribution of the different classes (in the case of 2, the positive/negative classes, for EP the rounded distribution of total votes in each class). $|W|$ is the average number of words per instance. We use EP≥ 4, a subset of entries with at least 4 votes.

4.1 EP Dataset: The Experience Project

The confessions section of the experience project website[3] lets people anonymously write short personal stories or "confessions". Once a story is on the site, each user can give a single vote to one of five label categories (with our interpretation):

1 Sorry, Hugs: User offers condolences to author.
2. You Rock: Indicating approval, congratulations.
3. Teehee: User found the anecdote amusing.
4. I Understand: Show of empathy.
5. Wow, Just Wow: Expression of surprise,shock.

The EP dataset has 31,676 confession entries, a total number of 74,859 votes for the 5 labels above, the average number of votes per entry is 2.4 (with a variance of 33). For the five categories, the numbers of votes are $[14,816; 13,325; 10,073; 30,844; 5,801]$. Since an entry with less than 4 votes is not very well identified, we train and test only on entries with at least 4 total votes. There are 6,129 total such entries. The distribution over total votes in the 5 classes is similar: $[0.22; 0.2; 0.11; 0.37; 0.1]$. The average length of entries is 129 words. Some entries contain multiple sentences. In these cases, we average the predicted label distributions from the sentences. Table 1 shows statistics of this and other commonly used sentiment datasets (which we compare on in later experiments). Table 2 shows example entries as well as gold and predicted label distributions as described in the next sections.

Compared to other datasets, the EP dataset contains a wider range of human emotions that goes far beyond positive/negative product or movie reviews. Each item is labeled with a multinomial distribution over interconnected response categories. This is in contrast to most other datasets (including multi-aspect rating) where several distinct aspects are rated independently but on the same scale. The topics range from generic happy statements, daily clumsiness reports, love, loneliness, to relationship abuse and suicidal notes. As is evident from the total number of label votes, the most common user reaction is one of empathy and an ability to relate to the authors experience. However, some stories describe horrible scenarios that are not common and hence receive more offers of condolence. In the following sections we show some examples of stories with predicted and true distributions but refrain from listing the most horrible experiences.

For all experiments on the EP dataset, we split the data into train (49%), development (21%) and test data (30%).

4.2 EP: Predicting the Label with Most Votes

The first task for our evaluation on the EP dataset is to simply predict the single class that receives the most votes. In order to compare our novel joint phrase representation and classifier learning framework to traditional methods, we use the following baselines:

Random Since there are five classes, this gives 20% accuracy.

Most Frequent Selecting the class which most frequently has the most votes (the class *I understand*).

Baseline 1: Binary BoW This baseline uses logistic regression on binary bag-of-word representations that are 1 if a word is present and 0 otherwise.

Baseline 2: Features This model is similar to traditional approaches to sentiment classification in that it uses many hand-engineered resources. We first used a spell-checker and Wordnet to map words and their misspellings to synsets to reduce the total number of words. We then replaced sentiment words with a sentiment category identifier using the sentiment lexica of the Harvard Inquirer (Stone, 1966) and LIWC (Pennebaker et al., 2007). Lastly, we used tf-idf weighting on the bag-of-word representations and trained an SVM.

[3] http://www.experienceproject.com/confessions.php

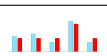
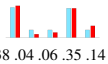
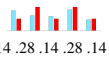
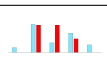
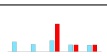
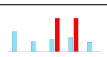
KL	Predicted&Gold	V.	Entry (Shortened if it ends with ...)
.03	 .16 .16 .16 .33 .16	6	I reguarly shoplift. I got caught once and went to jail, but I've found that this was not a deterrent. I don't buy groceries, I don't buy school supplies for my kids, I don't buy gifts for my kids, we don't pay for movies, and I dont buy most incidentals for the house (cleaning supplies, toothpaste, etc.)...
.03	 .38 .04 .06 .35 .14	165	i am a very succesfull buissnes man.i make good money but i have been addicted to crack for 13 years.i moved 1 hour away from my dealers 10 years ago to stop using now i dont use daily but once a week usally friday nights. i used to use 1 or 2 hundred a day now i use 4 or 5 hundred on a friday.my problem is i am a funcational addict...
.05	 .14 .28 .14 .28 .14	7	Hi there, Im a guy that loves a girl, the same old bloody story... I met her a while ago, while studying, she Is so perfect, so mature and yet so lonely, I get to know her and she get ahold of me, by opening her life to me and so did I with her, she has been the first person, male or female that has ever made that bond with me,...
.07	 .27 .18 .00 .45 .09	11	be kissing you right now. i should be wrapped in your arms in the dark, but instead i've ruined everything. i've piled bricks to make a wall where there never should have been one. i feel an ache that i shouldn't feel because i've never had you close enough. we've never touched, but i still feel as though a part of me is missing. ...
.05		23	Dear Love, I just want to say that I am looking for you. Tonight I felt the urge to write, and I am becoming more and more frustrated that I have not found you yet. I'm also tired of spending so much heart on an old dream. ...
.05		5	I wish I knew somone to talk to here.
.06		24	I loved her but I screwed it up. Now she's moved on. I'll never have her again. I don't know if I'll ever stop thinking about her.
.06		5	i am 13 years old and i hate my father he is alwas geting drunk and do's not care about how it affects me or my sisters i want to care but the truthis i dont care if he dies
.13		6	well i think hairy women are attractive
.35		5	As soon as I put clothings on I will go down to DQ and get a thin mint blizzard. I need it. It'll make my soul feel a bit better :)
.36		6	I am a 45 year old divoced woman, and I havent been on a date or had any significant relationship in 12 years...yes, 12 yrs. the sad thing is, Im not some dried up old granny who is no longer interested in men, I just can't meet men. (before you judge, no Im not terribly picky!) What is wrong with me?
.63		6	When i was in kindergarden i used to lock myself in the closet and eat all the candy. Then the teacher found out it was one of us and made us go two days without freetime. It might be a little late now, but sorry guys it was me haha
.92		4	My paper is due in less than 24 hours and I'm still dancing round my room!

Table 2: Example EP confessions from the test data with KL divergence between our predicted distribution (light blue, left bar on each of the 5 classes) and ground truth distribution (red bar and numbers underneath), number of votes. The 5 classes are [Sorry, Hugs ;You Rock; Teehee;I Understand;Wow, Just Wow]. Even when the KL divergence is higher, our model makes reasonable alternative label choices. Some entries are shortened.

Baseline 3: Word Vectors We can ignore the RAE tree structure and only train softmax layers directly on the pre-trained words in order to influence the word vectors. This is followed by an SVM trained on the average of the word vectors.

We also experimented with latent Dirichlet allocation (Blei et al., 2003) but performance was very low.

Table 3 shows the results for predicting the class with the most votes. Even the approach that is based on sentiment lexica and other resources is outperformed by our model by almost 3%, showing that for tasks involving complex broad-range human sentiment, the often used sentiment lexica lack in coverage and traditional bag-of-words representations are not powerful enough.

Method	Accuracy
Random	20.0
Most Frequent	38.1
Baseline 1: Binary BoW	46.4
Baseline 2: Features	47.0
Baseline 3: Word Vectors	45.5
RAE (our method)	**50.1**

Table 3: Accuracy of predicting the class with most votes.

4.3 EP: Predicting Sentiment Distributions

We now turn to evaluating our distribution-prediction approach. In both this and the previous maximum label task, we backprop using the gold multinomial distribution as a target. Since we maximize likelihood and because we want to predict a distribution that is closest to the distribution of labels that people would assign to a story, we evaluate using KL divergence: $KL(g||p) = \sum_i g_i \log(g_i/p_i)$, where g is the gold distribution and p is the predicted one. We report the average KL divergence, where a smaller value indicates better predictive power. To get an idea of the values of KL divergence, predict-

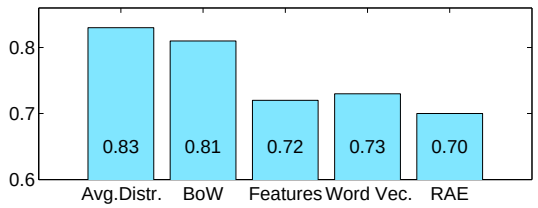


Figure 4: Average KL-divergence between gold and predicted sentiment distributions (lower is better).

ing random distributions gives a an average of 1.2 in KL divergence, predicting simply the average distribution in the training data give 0.83. Fig. 4 shows that our RAE-based model outperforms the other baselines. Table 2 shows EP example entries with predicted and gold distributions, as well as numbers of votes.

4.4 Binary Polarity Classification

In order to compare our approach to other methods we also show results on commonly used sentiment datasets: movie reviews[4] (MR) (Pang and Lee, 2005) and opinions[5] (MPQA) (Wiebe et al., 2005).We give statistical information on these and the EP corpus in Table 1.

We compare to the state-of-the-art system of (Nakagawa et al., 2010), a dependency tree based classification method that uses CRFs with hidden variables. We use the same training and testing regimen (10-fold cross validation) as well as their baselines: majority phrase voting using sentiment and reversal lexica; rule-based reversal using a dependency tree; Bag-of-Features and their full Tree-CRF model. As shown in Table 4, our algorithm outperforms their approach on both datasets. For the movie review (MR) data set, we do not use any hand-designed lexica. An error analysis on the MPQA dataset showed several cases of single words which never occurred in the training set. Correctly classifying these instances can only be the result of having them in the original sentiment lexicon. Hence, for the experiment on MPQA we added the same sentiment lexicon that (Nakagawa et al., 2010) used in their system to our training set. This improved accuracy from 86.0 to 86.4.Using the pre-trained word vectors boosts performance by less than 1% compared to randomly initialized word vectors (setting: random word init). This shows that our method can work well even in settings with little training data. We visualize the semantic vectors that the recursive autoencoder learns by listing n-grams that give the highest probability for each polarity. Table 5 shows such n-grams for different lengths when the RAE is trained on the movie review polarity dataset.

On a 4-core machine, training time for the smaller corpora such as the movie reviews takes around 3 hours and for the larger EP corpus around 12 hours until convergence. Testing of hundreds of movie reviews takes only a few seconds.

[4] www.cs.cornell.edu/people/pabo/movie-review-data/

[5] www.cs.pitt.edu/mpqa/

Method	MR	MPQA
Voting with two lexica	63.1	81.7
Rule-based reversal on trees	62.9	82.8
Bag of features with reversal	76.4	84.1
Tree-CRF (Nakagawa et al,'10)	77.3	86.1
RAE (random word init.)	76.8	85.7
RAE (our method)	**77.7**	**86.4**

Table 4: Accuracy of sentiment classification on movie review polarity (MR) and the MPQA dataset.

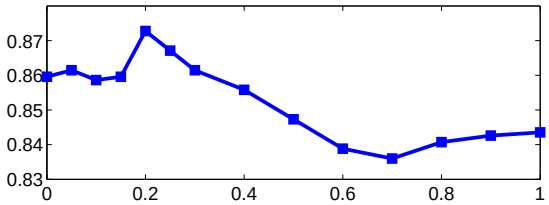


Figure 5: Accuracy on the development split of the MR polarity dataset for different weightings of reconstruction error and supervised cross-entropy error: $err = \alpha E_{rec} + (1-\alpha)E_{cE}$.

4.5 Reconstruction vs. Classification Error

In this experiment, we show how the hyperparameter α influences accuracy on the development set of one of the cross-validation splits of the MR dataset. This parameter essentially trade-off the supervised and unsupervised parts of the objective. Fig. 5 shows that a larger focus on the supervised objective is important but that a weight of $\alpha = 0.2$ for the reconstruction error prevents overfitting and achieves the highest performance.

n	Most negative n-grams	Most positive n-grams
1	bad; boring; dull; flat; pointless; tv; neither; pretentious; badly; worst; lame; mediocre; lack; routine; loud; bore; barely; stupid; tired; poorly; suffers; heavy;nor; choppy; superficial	touching; enjoyable; powerful; warm; moving; culture; flaws; provides; engrossing; wonderful; beautiful; quiet; socio-political; thoughtful; portrait; refreshingly; chilling; rich; beautifully; solid;
2	how bad; by bad; dull .; for bad; to bad; boring .; , dull; are bad; that bad; boring ,; , flat; pointless .; badly by; on tv; so routine; lack the; mediocre .; a generic; stupid ,; abysmally pathetic	the beautiful; moving,; thoughtful and; , inventive; solid and; a beautiful; a beautifully; and hilarious; with dazzling; provides the; provides.; and inventive; as powerful; moving and; a moving; a powerful
3	. too bad; exactly how bad; and never dull; shot but dull; is more boring; to the dull; dull, UNK; it is bad; or just plain; by turns pretentious; manipulative and contrived; bag of stale; is a bad; the whole mildly; contrived pastiche of; from this choppy; stale material.	funny and touching; a small gem; with a moving; cuts, fast; , fine music; smart and taut; culture into a; romantic , riveting; ... a solid; beautifully acted .; , gradually reveals; with the chilling; cast of solid; has a solid; spare yet audacious; ... a polished; both the beauty;
5	boring than anything else.; a major waste ... generic; nothing i hadn't already; ,UNK plotting;superficial; problem ? no laughs.; ,just horribly mediocre .; dull, UNK feel.; there's nothing exactly wrong; movie is about a boring; essentially a collection of bits	reminded us that a feel-good; engrossing, seldom UNK,; between realistic characters showing honest; a solid piece of journalistic; easily the most thoughtful fictional; cute, funny, heartwarming; with wry humor and genuine; engrossing and ultimately tragic.;
8	loud, silly, stupid and pointless.; dull, dumb and derivative horror film.; UNK's film, a boring, pretentious; this film biggest problem ? no laughs.; film in the series looks and feels tired; do draw easy chuckles but lead nowhere.; stupid, infantile, redundant, sloppy	shot in rich , shadowy black-and-white , devils an escapist confection that 's pure entertainment .; , deeply absorbing piece that works as a; ... one of the most ingenious and entertaining; film is a riveting , brisk delight .; bringing richer meaning to the story 's;

Table 5: Examples of n-grams ($n = 1, 2, 3, 5, 8$) from the test data of the movie polarity dataset for which our model predicts the most positive and most negative responses.

5 Related Work

5.1 Autoencoders and Deep Learning

Autoencoders are neural networks that learn a reduced dimensional representation of fixed-size inputs such as image patches or bag-of-word representations of text documents. They can be used to efficiently learn feature encodings which are useful for classification. Recently, Mirowski et al. (2010) learn dynamic autoencoders for documents in a bag-of-words format which, like ours, combine supervised and reconstruction objectives.

The idea of applying an autoencoder in a recursive setting was introduced by Pollack (1990). Pollack's recursive auto-associative memories (RAAMs) are similar to ours in that they are a connectionst, feed-forward model. However, RAAMs learn vector representations only for fixed recursive data structures, whereas our RAE builds this recursive data structure. More recently, (Voegtlin and Dominey, 2005) introduced a linear modification to RAAMs that is able to better generalize to novel combinations of previously seen constituents. One of the major shortcomings of previous applications of recursive autoencoders to natural language sentences was their binary word representation as discussed in Sec. 2.1.

Recently, (Socher et al., 2010; Socher et al., 2011) introduced a max-margin framework based on recursive neural networks (RNNs) for labeled structure prediction. Their models are applicable to natural language and computer vision tasks such as parsing or object detection. The current work is related in that it uses a recursive deep learning model. However, RNNs require labeled tree structures and use a supervised score at each node. Instead, RAEs learn hierarchical structures that are trying to capture as much of the the original word vectors as possible. The learned structures are not necessarily syntactically plausible but can capture more of the semantic content of the word vectors. Other recent deep learning methods for sentiment analysis include (Maas et al., 2011).

5.2 Sentiment Analysis

Pang et al. (2002) were one of the first to experiment with sentiment classification. They show that simple bag-of-words approaches based on Naive Bayes, MaxEnt models or SVMs are often insufficient for predicting sentiment of documents even though they work well for general topic-based document classification. Even adding specific negation words, bigrams or part-of-speech information to these models did not add significant improvements. Other document-level sentiment work includes (Turney, 2002; Dave et al., 2003; Beineke et al., 2004; Pang and Lee, 2004). For further references, see (Pang and Lee, 2008).

Instead of document level sentiment classification, (Wilson et al., 2005) analyze the contextual polarity of phrases and incorporate many well designed features including dependency trees. They also show improvements by first distinguishing be-

tween neutral and polar sentences. Our model naturally incorporates the recursive interaction between context and polarity words in sentences in a unified framework while simultaneously learning the necessary features to make accurate predictions. Other approaches for sentence-level sentiment detection include (Yu and Hatzivassiloglou, 2003; Grefenstette et al., 2004; Ikeda et al., 2008).

Most previous work is centered around a given sentiment lexicon or building one via heuristics (Kim and Hovy, 2007; Esuli and Sebastiani, 2007), manual annotation (Das and Chen, 2001) or machine learning techniques (Turney, 2002). In contrast, we do not require an initial or constructed sentiment lexicon of positive and negative words. In fact, when training our approach on documents or sentences, it jointly learns such lexica for both single words and n-grams (see Table 5). (Mao and Lebanon, 2007) propose isotonic conditional random fields and differentiate between local, sentence-level and global, document-level sentiment.

The work of (Polanyi and Zaenen, 2006; Choi and Cardie, 2008) focuses on manually constructing several lexica and rules for both polar words and related content-word negators, such as "prevent cancer", where *prevent* reverses the negative polarity of *cancer*. Like our approach they capture compositional semantics. However, our model does so without manually constructing any rules or lexica.

Recently, (Velikovich et al., 2010) showed how to use a seed lexicon and a graph propagation framework to learn a larger sentiment lexicon that also includes polar multi-word phrases such as "once in a life time". While our method can also learn multi-word phrases it does not require a seed set or a large web graph. (Nakagawa et al., 2010) introduced an approach based on CRFs with hidden variables with very good performance. We compare to their state-of-the-art system. We outperform them on the standard corpora that we tested on without requiring external systems such as POS taggers, dependency parsers and sentiment lexica. Our approach jointly learns the necessary features and tree structure.

In multi-aspect rating (Snyder and Barzilay, 2007) one finds several distinct aspects such as food or service in a restaurant and then rates them on a fixed linear scale such as 1-5 stars, where all aspects could obtain just 1 star or all aspects could obtain 5 stars independently. In contrast, in our method a single aspect (a complex reaction to a human experience) is predicted not in terms of a fixed scale but in terms of a *multinomial distribution* over several interconnected, sometimes mutually exclusive emotions. A single story cannot simultaneously obtain a strong reaction in different emotional responses (by virtue of having to sum to one).

6 Conclusion

We presented a novel algorithm that can accurately predict sentence-level sentiment distributions. Without using any hand-engineered resources such as sentiment lexica, parsers or sentiment shifting rules, our model achieves state-of-the-art performance on commonly used sentiment datasets. Furthermore, we introduce a new dataset that contains distributions over a broad range of human emotions. Our evaluation shows that our model can more accurately predict these distributions than other models.

Acknowledgments

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government. This work was also supported in part by the DARPA Deep Learning program under contract number FA8650-10-C-7020.
We thank Chris Potts for help with the EP data set, Raymond Hsu, Bozhi See, and Alan Wu for letting us use their system as a baseline and Jiquan Ngiam, Quoc Le, Gabor Angeli and Andrew Maas for their feedback.

References

P. Beineke, T. Hastie, C. D. Manning, and S. Vaithyanathan. 2004. Exploring sentiment summarization. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research.*, 3:993–1022.

Y. Choi and C. Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *EMNLP*.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

S. Das and M. Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific Finance Association Annual Conference (APFA)*.

K. Dave, S. Lawrence, and D. M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of WWW*, pages 519–528.

X. Ding, B. Liu, and P. S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the Conference on Web Search and Web Data Mining (WSDM)*.

J. L. Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2-3):195–225.

A. Esuli and F. Sebastiani. 2007. Pageranking wordnet synsets: An application to opinion mining. In *Proceedings of the Association for Computational Linguistics (ACL)*.

C. Goller and A. Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks (ICNN-96)*.

G. Grefenstette, Y. Qu, J. G. Shanahan, and D. A. Evans. 2004. Coupling niche browsers and affect analysis for an opinion mining application. In *Proceedings of Recherche d'Information Assistée par Ordinateur (RIAO)*.

D. Ikeda, H. Takamura, L. Ratinov, and M. Okumura. 2008. Learning to shift the polarity of words for sentiment classification. In *IJCNLP*.

S. Kim and E. Hovy. 2007. Crystal: Analyzing predictive opinions on the web. In *EMNLP-CoNLL*.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. 2011. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of ACL*.

Y. Mao and G. Lebanon. 2007. Isotonic Conditional Random Fields and Local Sentiment Flow. In *NIPS*.

P. Mirowski, M. Ranzato, and Y. LeCun. 2010. Dynamic auto-encoders for semantic indexing. In *Proceedings of the NIPS 2010 Workshop on Deep Learning*.

T. Nakagawa, K. Inui, and S. Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. In *NAACL*, HLT.

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

B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

B. Pang and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *EMNLP*.

J. W. Pennebaker, R.J. Booth, and M. E. Francis. 2007. Linguistic inquiry and word count: Liwc2007 operators manual. University of Texas.

L. Polanyi and A. Zaenen. 2006. Contextual valence shifters.

J. B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46:77–105, November.

C. Potts. 2010. On the negativity of negation. In David Lutz and Nan Li, editors, *Proceedings of Semantics and Linguistic Theory 20*. CLC Publications, Ithaca, NY.

B. Snyder and R. Barzilay. 2007. Multiple aspect ranking using the Good Grief algorithm. In *HLT-NAACL*.

R. Socher, C. D. Manning, and A. Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. 2011. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *ICML*.

P. J. Stone. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

P. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *ACL*.

L. Velikovich, S. Blair-Goldensohn, K. Hannan, and R. McDonald. 2010. The viability of web-derived polarity lexicons. In *NAACL*, HLT.

T. Voegtlin and P. Dominey. 2005. Linear Recursive Distributed Representations. *Neural Networks*, 18(7).

J. Wiebe, T. Wilson, and C. Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39.

T. Wilson, J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP*.

H. Yu and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *EMNLP*.

Unsupervised Discovery of Discourse Relations for Eliminating Intra-sentence Polarity Ambiguities

Lanjun Zhou, Binyang Li, Wei Gao, Zhongyu Wei, Kam-Fai Wong
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Shatin, NT, Hong Kong, China
Key Laboratory of High Confidence Software Technologies
Ministry of Education, China
{ljzhou, byli, wgao, zywei, kfwong}@se.cuhk.edu.hk

Abstract

Polarity classification of opinionated sentences with both positive and negative sentiments[1] is a key challenge in sentiment analysis. This paper presents a novel unsupervised method for discovering intra-sentence level discourse relations for eliminating polarity ambiguities. Firstly, a discourse scheme with discourse constraints on polarity was defined empirically based on Rhetorical Structure Theory (RST). Then, a small set of cue-phrase-based patterns were utilized to collect a large number of discourse instances which were later converted to semantic sequential representations (SSRs). Finally, an unsupervised method was adopted to generate, weigh and filter new SSRs without cue phrases for recognizing discourse relations. Experimental results showed that the proposed methods not only effectively recognized the defined discourse relations but also achieved significant improvement by integrating discourse information in sentence-level polarity classification.

1 Introduction

As an important task of sentiment analysis, polarity classification is critically affected by discourse structure (Polanyi and Zaenen, 2006). Previous research developed discourse schema (Asher et al., 2008) (Somasundaran et al., 2008) and proved that the utilization of discourse relations could improve the performance of polarity classification on dialogues (Somasundaran et al., 2009). However, current state-of-the-art methods for sentence-level polarity classification are facing difficulties in ascertaining the polarity of some sentences. For example:

(a) [Although Fujimori was criticized by the international community], [he was loved by the domestic population], [because people hated the corrupted ruling class]. (儘管國際間對藤森口誅筆伐，他在國內一直深受百姓愛戴，原因是百姓對腐化的統治階級早就深惡痛絕。)

Example (a) is a positive sentence holding a *Contrast* relation between first two segments and a *Cause* relation between last two segments. The polarity of "criticized", "hated" and "corrupted" are recognized as negative expressions while "loved" is recognized as a positive expression. Example (a) is difficult for existing polarity classification methods for two reasons: (1) the number of positive expressions is less than negative expressions; (2) the importance of each sentiment expression is unknown. However, consider Figure 1, if we know that the polarity of the first two segments holding a *Contrast* relation is determined by the *nucleus* (Mann and Thompson, 1988) segment and the polarity of the last two segments holding a *Cause* relation is also determined by the *nucleus* segment, the polarity of the sentence will be determined by the polarity of "[he...population]". Thus, the polarity of Example (a) is positive.

Statistics showed that 43% of the opinionated sentences in NTCIR[2] MOAT (Multilingual Opinion Analysis Task) Chinese corpus[3] are *ambiguous*. Existing sentence-level polarity classification methods ignoring discourse structure often give wrong results for these sentences. We implemented state-of-the-

[1] Defined as *ambiguous* sentences in this paper

[2] http://research.nii.ac.jp/ntcir/

[3] Including simplified Chinese and traditional Chinese corpus from NTCIR-6 MOAT and NTCIR-7 MOAT

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 162–171,
Edinburgh, Scotland, UK, July 27–31, 2011.

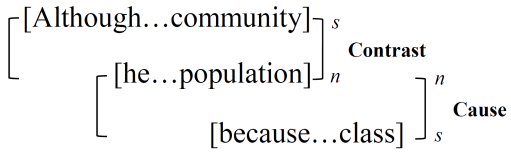


Figure 1: Discourse relations for Example (a). (*n* and *s* denote *nucleus* and *satellite* segment, respectively)

art method (Xu and Kit, 2010) in NTCIR-8 Chinese MOAT as the baseline polarity classifier (BPC) in this paper. Error analysis of BPC showed that 49% errors came from *ambiguous* sentences.

In this paper, we focused on the automation of recognizing intra-sentence level discourse relations for polarity classification. Based on the previous work of Rhetorical Structure Theory (RST) (Mann and Thompson, 1988), a discourse scheme with discourse constraints on polarity was defined empirically (see Section 3). The scheme contains 5 relations: *Contrast*, *Condition*, *Continuation*, *Cause* and *Purpose*. From a raw corpus, a small set of cue-phrase-based patterns were used to collect discourse instances. These instances were then converted to semantic sequential representations (SSRs). Finally, an unsupervised SSR learner was adopted to generate, weigh and filter high quality new SSRs without cue phrases. Experimental results showed that the proposed methods could effectively recognize the defined discourse relations and achieve significant improvement in sentence-level polarity classification comparing to BPC.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the discourse scheme with discourse constraints on polarity. Section 4 gives the detail of proposed method. Experimental results are reported and discussed in Section 5 and Section 6 concludes this paper.

2 Related Work

Research on polarity classification were generally conducted on 4 levels: document-level (Pang et al., 2002), sentence-level (Riloff et al., 2003), phrase-level (Wilson et al., 2009) and feature-level (Hu and Liu, 2004; Xia et al., 2007).

There was little research focusing on the automatic recognition of intra-sentence level discourse relations for sentiment analysis in the literature. Polanyi and Zaenen (2006) argued that valence calculation is critically affected by discourse structure. Asher et al. (2008) proposed a shallow semantic representation using a feature structure and use five types of rhetorical relations to build a fine-grained corpus for deep contextual sentiment analysis. Nevertheless, they did not propose a computational model for their discourse scheme. Snyder and Barzilay (2007) combined an agreement model based on contrastive RST relations with a local aspect model to make a more informed overall decision for sentiment classification. Nonetheless, contrastive relations were only one type of discourse relations which may help polarity classification. Sadamitsu et al. (2008) modeled polarity reversal using HCRFs integrated with inter-sentence discourse structures. However, our work is on intra-sentence level and our purpose is not to find polarity reversals but trying to adapt general discourse schemes (e.g., RST) to help determine the overall polarity of *ambiguous* sentences.

The most closely related works were (Somasundaran et al., 2008) and (Somasundaran et al., 2009), which proposed *opinion frames* as a representation of discourse-level associations on dialogue and modeled the scheme to improve opinion polarity classification. However, *opinion frames* was difficult to be implemented because the recognition of opinion target was very challenging in general text. Our work differs from their approaches in two key aspects: (1) we distinguished *nucleus* and *satellite* in discourse but *opinion frames* did not; (2) our method for discourse discovery was unsupervised while their method needed annotated data.

Most research works about discourse classification were not related to sentiment analysis. Supervised discourse classification methods (Soricut and Marcu, 2003; Duverle and Prendinger, 2009) needed manually annotated data. Marcu and Echihabi (2002) presented an unsupervised method to recognize discourse relations held between arbitrary spans of text. They showed that lexical pairs extracted from massive amount of data can have a major impact on discourse classification. Blair-Goldensohn et al. (2007) extended Marcu's work by using parameter opitimization, topic segmentation and syntactic parsing. However, syntactic parsers

were usually costly and impractical when dealing with large scale of text. Thus, in additional to lexical features, we incorporated sequential and semantic information in proposed method for discourse relation classification. Moreover, our method kept the characteristic of language independent, so it could be applied to other languages.

3 Discourse Scheme for Eliminating Polarity Ambiguities

Since not all of the discourse relations in RST would help eliminate polarity ambiguities, the discourse scheme defined in this paper was on a much coarser level. In order to ascertain which relations should be included in our scheme, 500 *ambiguous* sentences were randomly chosen from NTCIR MOAT Chinese corpus and the most common discourse relations for connecting independent clauses in compound sentences were annotated. We found that 13 relations from RST occupied about 70% of the annotated discourse relations which may help eliminate polarity ambiguities. Inspired by Marcu and Echihabi (2002), to construct relatively low-noise discourse instances for unsupervised methods using cue phrases, we grouped the 13 relations into the following 5 relations:

Contrast is a union of *Antithesis*, *Concession*, *Otherwise* and *Contrast* from RST.

Condition is selected from RST.

Continuation is a union of *Continuation*, *Parallel* from RST.

Cause is a union of *Evidence*, *Volitional-Cause*, *Nonvolitional-Cause*, *Volitional-result* and *Nonvolitional-result* from RST.

Purpose is selected from RST.

The discourse constraints on polarity presented here were based on the observation of annotated discourse instances: (1) discourse instances holding *Contrast* relation should contain two segments with opposite polarities; (2) discourse instances holding *Continuation* relation should contain two segments with the same polarity; (3) the polarity of discourse instances holding *Contrast*, *Condition*, *Cause* or *Purpose* was determined by the *nucleus* segment; (4) the polarity of discourse instances holding *Continuation* was determined by either segment.

Relation	Cue Phrases (English Translation)
Contrast	although[1], but[2], however[2]
Condition	if[1], (if[1], then[2])
Continuation	and, further more, (not only, but also)
Cause	because[1], thus[2], accordingly[2], as a result[2]
Purpose	in order to[2], in order that[2], so that[2]

[1] means *CUE1* and [2] means *CUE2*

Table 1: Examples of cue phrases

4 Methods

The proposed methods were based on two assumptions: (1) Cue-phrase-based patterns could be used to find limited number of high quality discourse instances; (2) discourse relations were determined by lexical, structural and semantic information between two segments.

Cue-phrase-based patterns could find only limited number of discourse instances with high precision (Marcu and Echihabi, 2002). Therefore, we could not rely on cue-phrase-based patterns alone. Moreover, there was no annotated corpus similar to Penn Discourse TreeBank (Miltsakaki et al., 2004) in other languages such as Chinese. Thus, we proposed a language independent unsupervised method to identify discourse relations without cue phrases while maintaining relatively high precision. For each discourse relation, we started with several cue-phrase-based patterns and collected a large number of discourse instances from raw corpus. Then, discourse instances were converted to semantic sequential representations (SSRs). Finally, an unsupervised method was adopted to generate, weigh and filter common SSRs without cue phrases. The mined common SSRs could be directly used in our SSR-based classifier in unsupervised manner or be employed as effective features for supervised methods.

4.1 Gathering and representing discourse instances

A discourse instance, denoted by D_i, consists of two successive segments $(D_{i[1]}, D_{i[2]})$ within a sentence. For example:

D_1: [*Although Boris is very brilliant at math*]$_s$, [*he*

BOS... , *[CUE2]*...EOS
BOS *[CUE1]*... , ...EOS
BOS... , *[CUE1]*...EOS
BOS *[CUE1]*... , *[CUE2]*...EOS

Table 2: Cue-phrase-based patterns. BOS and EOS denoted the beginning and end of two segments.

is a horrible teacher]$_n$
D_2: [*John is good at basketball*]$_s$, [*but he lacks team spirit*]$_n$

In D_1, "although" indicated the *satellite* section while in D_2, "but" indicated the *nucleus* section. Accordingly, different cue phrases may indicate different segment type. Table 1 listed some examples of cue phrases for each discourse relation. Some cue phrases were singleton (e.g. "although" and "as a result") and some were used as a pair (e.g. "not only, but also"). "*CUE1*" indicated *satellite* segments and "*CUE2*" indicated *nucleus* segments. Note that we did not distinguish *satellite* from *nucleus* for *Continuation* in this paper because the polarity could be determined by either segment.

Table 2 listed cue-phrase-based patterns for all relations. To simplify the problem of discourse segmentation, we split compound sentences into discourse segments using commas and semicolons. Although we collected discourse instances from compound sentences only, the number of instances for each discourse relation was large enough for the proposed unsupervised method. Note that we only collected instances containing at least one sentiment word in each segment.

In order to incorporate lexical and semantic information in our method, we represented each word in a discourse instance using a part-of-speech tag, a semantic label and a sentiment tag. Then, all discourse instances were converted to SSRs. The rules for converting were as follows:

(1) Cue phrases and punctuations were ingored. But the information of *nucleus*(n) and *satellite*(s) was preserved.

(2) Adverbs(*RB*) appearing in sentiment lexicon, verbs(V), adjectives(*JJ*) and nouns(*NN*) were represented by their part-of-speech (pos) tag with semantic label ($semlabel$) if available.

(3) Named entities (*NE*; *PER*: person name; *ORG*: organization), pronouns (*PRP*), and function words were represented by their corresponding named entity tags and part-of-speech tags, respectively.

(4) Added sentiment tag (P: Positive; N: Negative) to all sentiment words.

By applying above rules, the SSRs for D_1 and D_2 would be:

d_1: [*PER V|Ja01 RB|Ka01 JJ|Ee14|P IN NN|Dk03*]$_s$, [*PRP V|Ja01 DT JJ|Ga16|N NN|Ae13*]$_n$
d_2: [*PER V|Ja01 JJ|Ee14|P IN NN|Bp12*]$_s$, [*PRP V|He15|N NN|Di10 NN|Dd08*]$_n$

Refer to d_1 and d_2, "Boris" could match "John" in SSRs because they were converted to "*PER*" and they all appeared at the beginning of discourse instances. "*Ja01*", "*Ee14*" etc. were semantic labels from Chinese synonym list extended version (Che et al., 2010). There were similar resources in other languages such as Wordnet(Fellbaum, 1998) in English. The next problem became how to start from current SSRs and generate new SSRs for recognizing discourse relations without cue phrases.

4.2 Mining common SSRs

Recall assumption (2), in order to incorporate lexical, structural and semantic information for the similarity calculation of two SSRs holding the same discourse relation, three types of matches were defined for $\{(u,v)|u \in d_{i[k]}, v \in d_{j[k]}, k = 1,2\}$: (1)Full match: (i) $u = v$ or (ii) $u.pos = v.pos$ and $u.semlabel = v.semlabel$ or (iii) $u.pos = v.pos$ and u had a sentiment tag and v had a sentiment tag or (iv) $u.pos$ and $v.pos \in$ {*PRP, PER, ORG*} (2) Partial match: $u.pos = v.pos$ but not Full match; (3) Mismatch: $u.pos \neq v.pos$.

Generating common SSRs

Intuitively, a simple way of estimating the similarity between two SSRs was using the number of mismatches. Therefore, we utilized $match(d_i, d_j)$ where $i \neq j$, which integrated the three types of matches defined above to calculate the number of mismatches and generate common SSRs. Consider Table 3, in common SSRs, full matches were preserved, partial matches were replaced by part of speech tags and mismatches were replaced by '*'s. The common SSRs generated during the calculation of $match(d_i, d_j)$ consisted of two parts. The first part was generated by $d_{i[1]}$ and $d_{j[1]}$ and the second part was generated by $d_{i[2]}$ and $d_{j[2]}$. We stipulated

d_1	d_2	mis	$conf$	ssr
PER	*PER*	0	0	*PER*
V|Ja01	*V|Ja01*	0	0	*V|Ja01*
RB|Ka01		+1	-0.298	*
JJ|Ee14|P	*JJ|Ee14|P*	0	0	*JJ|Ee14|P*
IN	*IN*	0	0	*IN*
NN|Dk03	*NN|Bp12*	0	-0.50	*NN*
$conf(ssr_{[1]}) = -0.798$				
PRP	*PRP*	0	0	*PRP*
V|Ja01	*V|He15|N*	0	-0.50	*V*
DT		+1	-0.184	*
JJ|Ga16|N		+1	-1.0	*
NN|Ae13	*NN|Di10*	0	-0.50	*NN*
	NN|Dd08	+1	-1.0	*
$conf(ssr_{[2]}) = -3.184$				

Table 3: Calculation of $match(d_1, d_2)$. ssr denoted the common SSR between d_1 and d_2 , $conf(ssr_{[1]})$ and $conf(ssr_{[2]})$ denoted the confidence of ssr.

that d_i and d_j could generate a common SSR if and only if the orders of *nucleus* segment and *satellite* segment were the same.

In order to guarantee relatively high quality common SSRs, we empirically set the upper threshold of the number of mismatches as 0.5 (i.e., $\leq 1/2$ of the number of words in the generated SSR). It's not difficult to figure out that the number of mismatches generated in Table 3 satisfied this requirement. As a result, for each discourse relation r_n, a corresponding common SSR set S_n could be obtained by adopting $match(d_i, d_j)$ where $i \neq j$ for all discourse instances. An advantage of $match(d_1, d_2)$ was that the generated common SSRs preserved the sequential structure of original discourse instances. And common SSRs allows us to build high precision discourse classifiers (See Section 5).

Weighing and filtering common SSRs

A problem of $match(d_i, d_j)$ was that it ignored some important information by treating different mismatches equally. For example, the adverb "very" in "very brilliant" of D_1 was not important for discourse recognition. In other words, the number of mismatches in $match(d_i, d_j)$ could not precisely reflect the confidence of the generated common SSRs. Therefore, it was needed to weigh different mismatches for the confidence calculation of common SSRs.

Intuitively, if a partial match or a mismatch (denoted by u_m) occurred very frequently in the generation of common SSRs, the importance of u_m tends to diminish. Inspired by the *tf-idf* model, given $ssr_i \in S_n$, we utilized the following equation to estimate the weight (denoted by w_m) of u_m.

$$w_m = -uf_m \cdot \log\left(|S_n| / ssrf_m\right)$$

where uf_m denoted the frequency of u_m during the generation of ssr_i, $|S_n|$ denoted the size of S_n and $ssrf_m$ denoted the number of common SSRs in S_n containing u_m . All weights were normalized to $[-1, 0)$.

Nouns (except for named entities) and verbs were most representative words in discourse recognition (Marcu and Echihabi, 2002). In addition, adjectives and adverbs appearing in sentiment lexicons were important for polarity classification. Therefore, for these 4 kinds of words, we utilized -1.0 for a mismatch and -0.50 for a partial match.

As we had got the weights for all partial matches and mismatches, the confidence of $ssr_i \in S_n$ could be calculated using the cumulation of weights of partial matches and mismatches in $ssr_{i[1]}$ and $ssr_{i[2]}$. Recall Table 3, $conf(ssr_{[1]})$ and $conf(ssr_{[2]})$ represented the confidence scores of $match(d_{i[1]}, d_{j[1]})$ and $match(d_{i[2]}, d_{j[2]})$, respectively. In order to control the quantity and quality of mined SSRs, a threshold $minconf$ was introduced. ssr_i will be preserved if and only if $conf(ssr_{i[1]}) \geq minconf$ and $conf(ssr_{i[2]}) \geq minconf$. The value of $minconf$ was tuned using the development data.

Finally, we combined adjacent '*'s and preserved SSRs containing at least one notional word and at least two words in each segment to meet the demand of maintaining high precision (e.g., "[* *DT* *]", "[*PER* *]" will be dropped). Moreover, since many of the SSRs were duplicated, we ranked all the generated SSRs according to their occurrences and dropped those appearing only once in order to preserve common SSRs. At last, SSRs appearing in more than one common SSR set were removed for maintaining the uniqueness of each set. The common SSR set S_n for each discourse relation r_n could be directly used in SSR-based unsupervised classifiers or be employed as effective features in supervised methods.

Relation	Occurrence
Contrast	86 (8.2%)
Condition	27 (2.6%)
Continuation	445 (42.2%)
Cause	123 (11.7%)
Purpose	55 (5.2%)
Others	318 (30.2%)

Table 4: Distribution of discourse relations on NTC-7. *Others* represents discourse relations not included in our discourse scheme.

5 Experiments

5.1 Annotation work and Data

We extracted all compound sentences which may contain the defined discourse relations from opinionated sentences (neutral ones were dropped) of NTCIR7 MOAT simplified Chinese training data. 1,225 discourse instances were extracted and two annotators were trained to annotate discourse relations according to the discourse scheme defined in Section 3. Note that we annotate both explicit and implicit discourse relations. The overall inter annotator agreement was 86.05% and the *Kappa-value* was 0.8031. Table 4 showed the distribution of annotated discourse relations based on the inter-annotator agreement. The proportion of occurrences of each discourse relations varied greatly. For example, *Continuation* was the most common relation in annotated corpus, but the occurrences of *Condition* relation were rare.

The experiments of this paper were performed using the following data sets:

NTC-7 contained manually annotated discourse instances (shown in Table 4). The experiments of discourse identification were performed on this data set.

NTC-8 contained all opinionated sentences (neutral ones were dropped) extracted from NTCIR8 MOAT simplified Chinese test data. The experiments of polarity ambiguity elimination using the identified discourse relations were performed on this data set.

XINHUA contained simplified Chinese raw news text from Xinhua.com (2002-2005). A word segmentation tool, a part-of-speech tagging tool, a named entity recognizer and a word sense disambiguation tool (Che et al., 2010) were adopted to all sentences. The common SSRs were mined from this data set.

5.2 Experimental Settings

Discourse relation identification

In order to systematically justify the effectiveness of proposed unsupervised method, following experiments were performed on NTC-7:

Baseline used only cue-phrase-based patterns.

M&E proposed by Marcu and Echihabi (2002). Given a discourse instance D_i, the probabilities: $P(r_k|(D_{i[1]}, D_{i[2]}))$ for each relation r_k were estimated on all text from XINHUA. Then, the most likely discourse relation was determined by taking the maximum over $argmax_k\{P(r_k|(D_{i[1]}, D_{i[2]})\}$.

cSSR used both cue-phrase-based patterns together with common SSRs for recognizing discourse relations. Common SSRs were mined from discourse instances extracted from XINHUA using cue-phrase-based patterns. Development data were randomly selected for tuning $minconf$.

SVM was trained utilizing cue phrases, probabilities from *M&E*, topic similarity, structure overlap, polarity of segments and mined common SSRs (Optional). The parameters of the SVM classifier were set by a grid search on the training set. We performed 4-fold cross validation on NTC-7 to get an average performance.

The purposes of introducing *SVM* in our experiment were: (1) to compare the performance of *cSSR* to supervised method; (2) to examine the effectiveness of integrating common SSRs as features for supervised methods.

Polarity ambiguity elimination

BPC was trained mainly utilizing punctuation, uni-gram, bi-gram features with confidence score output. Discourse classifiers such as *Baseline*, *cSSR* or *SVM* were adopted individually for the post-processing of BPC. Given an *ambiguous* sentence which contained more than one segment, an intuitive three-step method was adopted to integrated a discourse classifier and discourse constraints on polarity for the post-processing of BPC:

(1) Recognize all discourse relations together with *nucleus* and *satellite* information using a discourse classifier. The *nucleus* and *satellite* information is

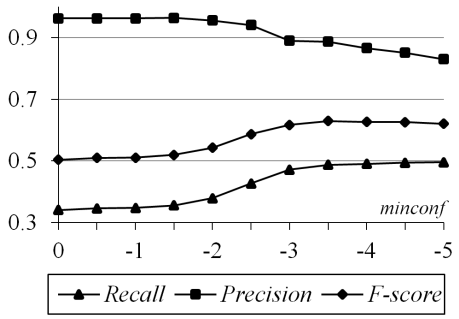


Figure 2: Influences of different values of $minconf$ to the performance of cSSR

acquired by cSSR if a segment pair could match a cSSR. Otherwise, we use the annotated *nucleus* and *satellite* information.

(2) Apply discourse constraints on polarity to ascertain the polarity for each discourse instance. There may be conflicts between polarities acquired by BPC and discourse constraints on polarity (e.g., Two segments with the same polarity holding a *Contrast* relation). To handle this problem, we chose the segment with higher polarity confidence and adjusted the polarity of the other segment using discourse constraints on polarity.

(3) If there was more than one discourse instance in a single sentence, the overall polarity of the sentence was determined by voting of polarities from each discourse instance under the majority rule.

5.3 Experimental Results

Refer to Figure 2, the performance of *cSSR* was significantly affected by $minconf$. Note that we performed the tuning process of $minconf$ on different development data ($1/4$ instances randomly selected from NTC-7) and Figure 2 showed the average performance. *cSSR* became *Baseline* when $minconf = 0$. A significant drop of precision was observed when $minconf$ was less than -2.5. The recall remained around 0.495 when $minconf \leq -4.0$. The best performance was observed when $minconf = -3.5$. As a result, -3.5 was utilized as the threshold value for *cSSR* in the following experiments.

Table 5 presented the experimental results for discourse relation classification. it showed that:

(1) Cue-phrase-based patterns could find only limited number of discourse relations (34.1% of average recall) with a very high precision (96.17% of average precision). This is a proof of assumption (1) given in Section 4. On the other side, *M&E* which only considered word pairs between two segments of discourse instances got a higher recall with a large drop of precision. The drop of precision may be caused by the neglect of structural and semantic information of discourse instances. However, *M&E* still outperformed *Baseline* in average $F\text{-}score$.

(2) *cSSR* enhanced *Baseline* by increasing the average recall by about 15% with only a small drop of precision. The performance of *cSSR* demonstrated that our method could effectively discover high quality common SSRs. The most remarkable improvement was observed on *Continuation* in which the recall increased by almost 20% with only a minor drop of precision. Actually, *cSSR* outperformed *Baseline* in all discourse relations except for *Contrast*. In Discourse Tree Bank (Carlson et al., 2001) only 26% of *Contrast* relations were indicated by cue phrases while in NTC-7 about 70% of *Contrast* were indicated by cue phrases. A possible reason was that we were dealing with Chinese news text which were usually well written. Another important observation was that the performance of *cSSR* was very close to the result of *SVM*.

(3) *SVM+SSRs* achieved the best $F\text{-}score$ on *Continuation* and average performance. The integration of SSRs to the feature set of *SVM* contributed to a remarkable increase in average $F\text{-}score$. The results of *cSSR* and *SVM+SSRs* demonstrated the effectiveness of common SSRs mined by the proposed unsupervised method.

Table 6 presented the performance of integrating discourse classifiers to polarity classification. For *Baseline* and *cSSR*, the information of *nucleus* and *satellite* could be obtained directly from cue-

	BPC	*Baseline*	*cSSR*	*SVM* +SSRs
Precision	0.7661	0.7982	0.8059	**0.8113**
Recall	0.7634	0.7957	0.8038	**0.8091**
F-score	0.7648	0.7970	0.8048	**0.8102**

Table 6: Performance of integrating discourse classifiers and constraints to polarity classification. Note that the experiments were performed on NTC-8 which contained only opinionated sentences.

Relation		*Baseline*	*M&E*	*cSSR*	*SVM*	*SVM* +SSRs
Contrast	**P**	**0.9375**	0.4527	0.7531	**0.9375**	**0.9375**
	R	0.6977	**0.7791**	0.7093	0.6977	0.6977
	F	**0.8000**	0.5726	0.7305	**0.8000**	**0.8000**
Condition	**P**	**1.0000**	0.4444	0.6774	**1.0000**	0.7083
	R	0.5556	**0.8889**	0.7778	0.5185	0.6296
	F	0.7143	0.5926	**0.7241**	0.6829	0.6667
Continuation	**P**	**0.9831**	0.6028	0.9761	0.6507	0.7266
	R	0.2607	0.5865	0.4584	**0.6697**	0.6629
	F	0.4120	0.5945	0.6239	0.6600	**0.6933**
Cause	**P**	**1.0000**	0.5542	0.9429	**1.0000**	0.9412
	R	0.2114	**0.3740**	0.2683	0.2114	0.2602
	F	0.3489	**0.4466**	0.4177	0.3489	0.4076
Purpose	**P**	0.8947	0.3704	0.8163	**0.9167**	0.7193
	R	0.6182	0.7273	0.7273	0.6000	**0.7455**
	F	0.7312	0.4908	**0.7692**	0.7253	0.7321
Average	**P**	**0.9617**	0.5302	0.8864	0.7207	0.7607
	R	0.3410	0.5951	0.4878	0.5856	**0.6046**
	F	0.5035	0.5608	0.6293	0.6461	**0.6737**

Table 5: Performance of recognizing discourse relations. (The evaluation criteria are **P**recision, **R**ecall and **F**-score)

phrase-based patterns and SSRs, respectively. For *SVM+cSSR*, the *nucleus* and *satellite* information was acquired by cSSR if a segment pair could match a cSSR. Otherwise, we used manually annotated *nucleus* and *satellite* information. It's clear that the performance of polarity classification was enhanced with the improvement of discourse relation recognition. *M&E* was not included in this experiment because the performance of polarity classification was decreased by the mis-classified discourse relations. *SVM+SSRs* achieved significant (p<0.01) improvement in polarity classification compared to BPC.

5.4 Discussion

Effect of weighing and filtering

To assess the contribution of weighing and filtering in mining SSRs using a minimum confidence threshold, i.e. $minconf$, we implemented *cSSR'* without weighing and filtering on the same data set. Consider Table 7, *cSSR* achieved obvious improvement in $Precision$ and $F\text{-}score$ than *cSSR'*. Moreover, the total number of SSRs was greatly reduced in *cSSR* with only a minor drop of recall. This was because *cSSR'* was affected by thousands of low quality common SSRs which would be filtered in *cSSR*. The result in Table 7 proved that weighing and filtering were essential in our proposed method.

	cSSR'	*cSSR*
Precision	0.6182	**0.8864**
Recall	**0.5014**	0.4878
F-score	0.5537	**0.6293**
NOS	$>$ 1 million	$\approx$ 0.12 million

Table 7: Comparison of *cSSR'* and *cSSR*. "NOS" denoted the number of mined common SSRs.

We further analyzed how the improvement was achieved in *cSSR*. In our experiment, the most common mismatches were auxiliary words, named entities, adjectives or adverbs without sentiments (e.g., "green", "very", etc.), prepositions, numbers and quantifiers. It's straightforward that these words were insignificant in discourse relation classification purpose. Moreover, these words did not belong to the 4 kinds of most representative words. In other words, the weights of most mismatches were calculated using the equation presented in Section 4.2 instead of utilizing a unified value, i.e. -1. Recall Table 3, the weight of "*RB|Ka01*" (original: "very") was -0.298 and "*DT*" (original: 'a') was -0.184. Comparing to the weights of mismatches for most representative words (-1.0), the proposed method successfully down weighed the words which were

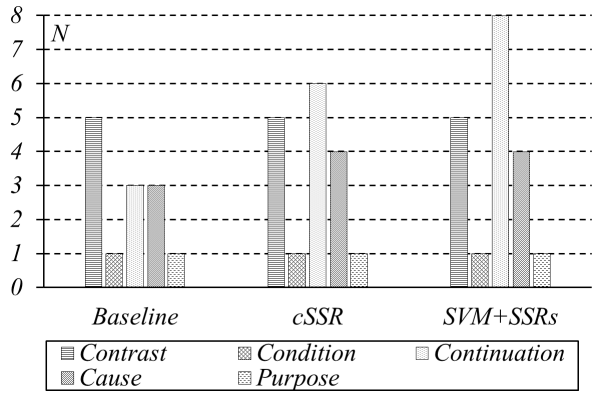


Figure 3: Improvement from individual discourse relations. N denoted the number of ambiguities eliminated.

not important for discourse identification. Therefore, weighing and filtering were able to preserve high quality SSRs while filter out low quality SSRs by setting the confidence threshold, i.e. $minconf$.

Contribution of different discourse relations

We also analyzed the contribution of different discourse relations in eliminating polarity ambiguities. Refer to Figure 3, the improvement of polarity classification mainly came from three discourse relations: *Contrast*, *Continuation* and *Cause*. It was straightforward that *Contrast* relation could eliminate polarity ambiguities because it held between two segments with opposite polarities. The contribution of *Cause* relation also result from two segments holding different polarities such as example (a) in Section 1. However, recall Table 4, although *Cause* occurred more often than *Contrast*, only a part of discourse instances holding *Cause* relation contained two segments with the opposite polarities. Another important relation in eliminating ambiguity was *Continuation*. We investigated sentences with polarities corrected by *Continuation* relation. Most of them fell into two categories: (1) sentences with mistakenly classified sentiments by BPC; (2) sentences with implicit sentiments. For example:

(b) [France and Germany have banned human cloning at present]，[on 20th, U.S. President George W. Bush called for regulations of the same content to Congress] (目前，法国和德国都禁止克隆人的胚胎，美国总统布什 20 日向国会提出，要求制定同样内容的法规。)

The first segment of example (b) was negative ("banned" expressed a negative sentiment) and a *Continuation* relation held between these two segments. Consequently, the polarity of the second segment should be negative.

6 Conclusions and Future work

This paper focused on unsupervised discovery of intra-sentence discourse relations for sentence level polarity classification. We firstly presented a discourse scheme based on empirical observations. Then, an unsupervised method was proposed starting from a small set of cue-phrase-based patterns to mine high quality common SSRs for each discourse relation. The performance of discourse classification was further improved by employing SSRs as features in supervised methods. Experimental results showed that our methods not only effectively recognized discourse relations but also achieved significant improvement ($p<0.01$) in sentence level polarity classification. Although we were dealing with Chinese text, the proposed unsupervised method could be easily generalized to other languages.

The future work will be focused on (1) integrating more semantic and syntactic information in proposed unsupervised method; (2) extending our method to inter-sentence level and then jointly modeling intra-sentence level and inter-sentence level discourse constraints on polarity to reach a global optimal inference for polarity classification.

Acknowledgments

This work is partially supported by National 863 program of China (Grant No. 2009AA01Z150), the Innovation and Technology Fund of Hong Kong SAR (Project No. GHP/036/09SZ) and 2010/11 CUHK Direct Grants (Project No. EE09743).

References

N. Asher, F. Benamara, and Y.Y. Mathieu. 2008. Distilling opinion in discourse: A preliminary study. *Coling 2008: Companion volume: Posters and Demonstrations*, pages 5--8.

S. Blair-Goldensohn, K.R. McKeown, and O.C. Rambow. 2007. Building and refining rhetorical-semantic relation models. In *Proceedings of NAACL HLT*, pages 428--435.

L. Carlson, D. Marcu, and M.E. Okurowski. 2001. Building a discourse-tagged corpus in the framework of

rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue-Volume 16*, pages 1--10. Association for Computational Linguistics.

W. Che, Z. Li, and T. Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13--16. Association for Computational Linguistics.

D.A. Duverle and H. Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 665--673. Association for Computational Linguistics.

C. Fellbaum. 1998. *WordNet: An electronic lexical database*. The MIT press.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168--177. ACM.

W.C. Mann and S.A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243--281.

D. Marcu and A. Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 368--375. Association for Computational Linguistics.

E. Miltsakaki, R. Prasad, A. Joshi, and B. Webber. 2004. The penn discourse treebank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*. Citeseer.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79--86. Association for Computational Linguistics.

L. Polanyi and A. Zaenen. 2006. Contextual valence shifters. *Computing attitude and affect in text: Theory and applications*, pages 1--10.

E. Riloff, J. Wiebe, and T. Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25--32. Association for Computational Linguistics.

K. Sadamitsu, S. Sekine, and M. Yamamoto. 2008. Sentiment analysis based on probabilistic models using inter-sentence information.

B. Snyder and R. Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *Proceedings of NAACL HLT*, pages 300--307.

S. Somasundaran, J. Wiebe, and J. Ruppenhofer. 2008. Discourse level opinion interpretation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 801--808. Association for Computational Linguistics.

S. Somasundaran, G. Namata, J. Wiebe, and L. Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 170--179. Association for Computational Linguistics.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 149--156. Association for Computational Linguistics.

T. Wilson, J. Wiebe, and P. Hoffmann. 2009. Recognizing Contextual Polarity: an exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3):399--433.

Y.Q. Xia, R.F. Xu, K.F. Wong, and F. Zheng. 2007. The unified collocation framework for opinion mining. In *International Conference on Machine Learning and Cybernetics*, volume 2, pages 844--850. IEEE.

R. Xu and C. Kit. 2010. Incorporating feature-based and similarity-based opinion mining--ctl in ntcir-8 moat. In *Proceedings of the 8th NTCIR Workshop*, pages 276--281.

Compositional Matrix-Space Models for Sentiment Analysis

Ainur Yessenalina
Dept. of Computer Science
Cornell University
Ithaca, NY, 14853
ainur@cs.cornell.edu

Claire Cardie
Dept. of Computer Science
Cornell University
Ithaca, NY, 14853
cardie@cs.cornell.edu

Abstract

We present a general learning-based approach for phrase-level sentiment analysis that adopts an ordinal sentiment scale and is explicitly compositional in nature. Thus, we can model the compositional effects required for accurate assignment of phrase-level sentiment. For example, combining an adverb (e.g., "very") with a positive polar adjective (e.g., "good") produces a phrase ("very good") with increased polarity over the adjective alone. Inspired by recent work on distributional approaches to compositionality, we model each word as a matrix and combine words using iterated matrix multiplication, which allows for the modeling of both additive and multiplicative semantic effects. Although the multiplication-based matrix-space framework has been shown to be a theoretically elegant way to model composition (Rudolph and Giesbrecht, 2010), training such models has to be done carefully: the optimization is non-convex and requires a good initial starting point. This paper presents the first such algorithm for *learning* a matrix-space model for semantic composition. In the context of the phrase-level sentiment analysis task, our experimental results show statistically significant improvements in performance over a bag-of-words model.

1 Introduction

Sentiment analysis has been an active research area in recent years. Work in the area ranges from identifying the sentiment of individual words to determining the sentiment of phrases, sentences and documents (see Pang and Lee (2008) for a survey). The bulk of previous research, however, models just positive vs. negative sentiment, collapsing positive (or negative) words, phrases and documents of differing intensities into just one positive (or negative) class. For word-level sentiment, therefore, these methods would not recognize a difference in sentiment between words like "good" and "great", which have the same direction of polarity (i.e., positive) but different intensities. At the phrase level, the methods will fail to register compositional effects in sentiment brought about by intensifiers like "very", "absolutely", "extremely", etc. "Happy" and "very happy", for example, will both be considered simply "positive" in sentiment. In real-world settings, on the other hand, sentiment values extend across a polarity spectrum — from very negative, to neutral, to very positive. Recent research has shown, in particular, that modeling intensity at the phrase level is important for real-world natural language processing tasks including question answering and textual entailment (de Marneffe et al., 2010).

This paper describes a general approach for phrase-level sentiment analysis that takes these real-world requirements into account: *we adopt a five-level ordinal sentiment scale and present a learning-based method that assigns ordinal sentiment scores to phrases*.

Importantly, our approach will also be explicitly *compositional*[1] in nature so that it can accurately account for critical interactions among the words in

[1]The *Principle of Compositionality* asserts that the meaning of a complex expression is a function of the meanings of its constituent expressions and the rules used to combine them.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 172–182,
Edinburgh, Scotland, UK, July 27–31, 2011.

each sentiment-bearing phrase. Consider, for example, combining an adverb like "very" with a polar adjective like "good". "Good" has an *a priori* positive sentiment, so "very good" should be considered **more** positive even though "very", on its own, does not bear sentiment. Combining "very" with a negative adjective, like "bad", produces a phrase ("very bad") that should be characterized as more negative than the original adjective. Thus, it is convenient to think of the effect of combining an intensifying adverb with a polar adjective as being *multiplicative* in nature, if we assume the adjectives ("good" and "bad") to have positive and a negative sentiment scores, respectively.

Next, let us consider adverbial negators like "not" combined with polar adjectives. When modeling only positive and negative labels for sentiment, negators are generally treated as flipping the polarity of the adjective it modifies (Choi and Cardie, 2008; Nakagawa et al., 2010). However, recent work (Taboada et al., 2011; Liu and Seneff, 2009) suggests that the effect of the negator when ordinal sentiment scores are employed is more akin to dampening the adjective's polarity rather than flipping it. For example, if "perfect" has a strong positive sentiment, then the phrase "not perfect" is still positive, though to a lesser degree. And while "not terrible" is still negative, it is less negative than "terrible". For these cases, it is convenient to view "not" as shifting polarity to the opposite side of polarity scale by some value.

There are, of course, more interesting examples of compositional semantic effects on sentiment: e.g., *prevent cancer*, *ease the burden*. Here, the verbs *prevent* and *ease* act as content-word negators (Choi and Cardie, 2008) in that they modify the negative sentiment of their direct object arguments so that the phrase as a whole is perceived as somewhat positive.

Nonetheless, the vast majority of methods for phrase- and sentence-level sentiment analysis do not tackle the task compositionally: they, instead, employ a bag-of-words representation and, at best, incorporate additional features to account for negators, intensifiers, and for contextual valence shifters, which can change the sentiment over neighboring words (e.g., Polanyi and Zaenen (2004), Wilson et al. (2005) , Kennedy and Inkpen (2006), Shaikh et al. (2007)).

One notable exception is Moilanen and Pulman (2007), who propose a compositional semantic approach to assign a positive or negative sentiment to newspaper article titles. However, their knowledge-based approach presupposes the existence of a sentiment lexicon and a set of symbolic compositional rules.

But learning-based compositional approaches for sentiment analyis also exist. Choi and Cardie (2008), for example, propose an algorithm for phrase-based sentiment analysis that learns proper assignments of intermediate sentiment analysis decision variables given the *a priori* (i.e., out of context) polarity of the words in the phrase and the (correct) phrase-level polarity. As in Moilianen and Pulman (2007), semantic inference is based on (a small set of) hand-written compositional rules. In contrast, Nakagawa et. al (2010) use a dependency parse tree to guide the learning of compositional effects. Each of the above, however, uses a binary rather than an ordinal sentiment scale.

In contrast, our proposed method for phrase-level sentiment analysis is inspired by recent work on distributional approaches to compositionality. In particular, Baroni and Zamparelli (2010) tackle adjective-noun compositions using a vector representation for nouns and *learning* a matrix representation for each adjective. The adjective matrices are then applied as functions over the meanings of nouns — via matrix-vector multiplication — to derive the meaning of adjective-noun combinations. Rudolph and Giesbrecht (2010) show theoretically, that multiplicative matrix-space models are a general case of vector-space models and furthermore exhibit desirable properties for semantic analysis: they take into account word order and are algebraically, neurologically and psychologically plausible. This work, however, does not present an algorithm for learning such models; nor does it provide empirical evidence in favor of matrix-space models over vector-space models.

In the sections below, we propose a *learning-based* approach to assign *ordinal sentiment scores* to sentiment-bearing phrases using a general *compositional matrix-space model of language*. In contrast to previous work, all words are modeled as matrices, independent of their part-of-speech, and compositional inference is uniformly modeled as ma-

trix multiplication. To predict an ordinal scale sentiment value, we employ Ordered Logistic Regression, introducing a novel training algorithm to accommodate our compositional matrix-space representations (Section 2). To our knowledge, this is the first such algorithm for learning matrix-space models for semantic composition. We evaluate the approach on a standard sentiment corpus (Wiebe et al., 2005) (Section 3), making use of its manually annotated phrase-level annotations for polarity and intensity, and compare our approach to the more commonly employed bag-of-words model. We show (Section 4) that our matrix-space model significantly outperforms a bag-of-words model for the ordinal scale sentiment prediction task.

2 The Model for Ordinal Scale Sentiment Prediction

As described above, our task is to predict an ordinal scale sentiment value for a phrase. To this end, we employ a sentiment scale with five ordinal values: VERY NEGATIVE, NEGATIVE, NEUTRAL, POSITIVE and VERY POSITIVE. Given a set of phrase-level training examples with their gold-standard ordinal sentiment value, we then use an Ordered Logistic Regression (OLogReg) model for prediction. Unfortunately, our matrix-space representation precludes doing this directly.

We have chosen OLogReg, as opposed to say PRanking (Crammer and Singer, 2001), because optimization of the former is more attractive: the objective (likelihood) is smooth and the gradients are continuous. As will become clear shortly, learning our models is not trivial and it is important to use sophisticated off-the-shelf optimizers such as L-BFGS.

For a bag-of-words model, OLogReg learns one weight for each word and a set of thresholds by maximizing the likelihood of the training data. Typically, this is accomplished by using an optimizer like L-BFGS whose interface needs the value and gradient of the likelihood with respect to the parameters at their current values. In the next subsections, we instantiate OLogReg for our sentiment prediction task using a matrix-space word model (2.1 and 2.2) and a bag-of-words model (2.3). The learning formulation of bag-of-words OLogReg is convex therefore we will get the global optimum; in contrast, the optimization problem for matrix-space model is non-convex, it is important to initialize the model well. Initialization of the matrix-space model is discussed in Section 2.4.

2.1 Notation

In the subsequent subsections we will use the following notation. Let n be the number of phrases in the training set and let d be the number of words in the dictionary. Let x^i be the i-th phrase and y^i would be the label of x^i, where y^i takes r different values $y^i \in \{0, \ldots, r-1\}$. Then $|x^i|$ will denote the length of the phrase x^i, and the words in i-th phrase are: $x^i = x^i_1, x^i_2, \ldots, x^i_{|x^i|}$; $x^i_j, 1 \leq j \leq |x^i|$ is the j-th word of i-th phrase; where x^i_j is from the dictionary: $1 \leq x^i_j \leq d$.

In the case of the bag-of-words model, $\Phi(x^i) \in \mathbb{R}^d$ is the representation of the i-th phrase. $\Phi_j(x^i)$ counts the number of times the j-th word from the dictionary appears in the i-th phrase. Given a $w \in \mathbb{R}^d$ it assigns a score ξ_i to a phrase x^i by

$$\xi_i = w^T \Phi(x^i) = \sum_{j=1}^{|x^i|} w_{x^i_j} \tag{1}$$

In the case of the matrix-space model the $\Phi(x^i) \in \mathbb{R}^{|x^i| \times d}$ is the representation of the i-th phrase. $\Phi_{jk}(x^i)$ is 1, if x^i_j is the k-th word in the dictionary, and zero otherwise. Given $u, v \in \mathbb{R}^m$ and a set of matrices $\{W_p \in \mathbb{R}^{m \times m}\}_{p=1}^d$, one for each word, it assigns a score ξ_i to a phrase x^i by

$$\begin{aligned} \xi_i &= u^T \left(\prod_{j=1}^{|x^i|} \sum_{k=1}^{d} W_k \Phi_{jk}(x^i) \right) v \\ &= u^T \left(\prod_{j=1}^{|x^i|} W_{x^i_j} \right) v \end{aligned} \tag{2}$$

where $\prod_{j=1}^{|x^i|} W_{x^i_j} = W_{x^i_1} W_{x^i_2} \cdots W_{x^i_{|x^i|}}$ in **exactly this order**. We choose to map matrices to the real numbers by using vectors u and v from $\mathbb{R}^{m \times 1}$; so that $\xi = u^T M v$, where $M \in \mathbb{R}^{m \times m}$, which is sensitive to the order of matrices[2] , i.e. $u^T M_1 M_2 v \neq$

[2]Care must be taken in choosing way to map matrix to a real

$u^T M_2 M_1 v$.

Modeling composition. A $m \times m$ matrix, representing a word, can be considered as a linear function, mapping from $\mathbb{R}^m$ to $\mathbb{R}^m$. Composition of words is modeled by function composition, in our case composition of linear functions, i.e. matrix multiplication. Note, that unlike bag-of-words model, the matrix-space model takes word order into account, since matrix multiplication is not commutative operation.

2.2 Ordered Logistic Regression

Now we will describe our objective function for OLogReg and its derivatives. OLogReg has $r-1$ thresholds ($\kappa_0, \dots \kappa_{r-2}$), so introducing $\kappa_{-1} = -\infty$ and $\kappa_{r-1} = \infty$ leads to the unified expression for posterior probabilities for all values of k:

$$\begin{aligned} P(y^i = k|x) &= P(\kappa_{k-1} < \xi_i \leq \kappa_k) \\ &= F(\kappa_k - \xi_i) - F(\kappa_{k-1} - \xi_i) \end{aligned}$$

$F(x)$ is an inverse-logit function

$$F(x) = \frac{e^x}{1+e^x}$$

this is its derivative:

$$\frac{dF(x)}{dx} = F(x)(1 - F(x))$$

Therefore the negative loglikelihood of the training data will look like the following (Hardin and Hilbe, 2007):

$$L = -\sum_{i=1}^{n}\sum_{k=0}^{r-1} \ln(F(\kappa_k - \xi_i) - F(\kappa_{k-1} - \xi_i))I(y^i = k)$$

where r is the number of ordinal classes, ξ_i is the score of i-th phrase, I is the indicator function that is equal to 1 – when $y^i = k$, and zero otherwise. We need to minimize the objective L with respect to the following constraints:

$$\kappa_{k-1} \leq \kappa_k, \;\; 1 \leq k \leq r-2 \tag{3}$$

number. For example, one other way to map matrices to the real numbers is to use the determinant of a matrix; however, the determinant is not sensitive to the word order: $det(M_1 M_2) = det(M_1)det(M_2) = det(M_2 M_1)$; which is not desirable for a model that needs to account for word order.

(The constraints are similar to the ones in PRank algorithm). For ease of optimization we parametrize our model via κ_0, and $\tau_j, 1 \leq j \leq r-2$:

$\kappa_{-1} = -\infty$,
κ_0,
$\kappa_1 = \kappa_0 + \tau_1$,
$\kappa_2 = \kappa_0 + \sum_{j=1}^{2} \tau_j$,
$\dots$,
$\kappa_{r-2} = \kappa_0 + \sum_{j=1}^{r-2} \tau_j$
$\kappa_{r-1} = \infty$,

where $\tau_1, \dots, \tau_{r-2}$ are non-negative values, that represent how far the corresponding thresholds are from each other. Then the constraints (3) would be:

$$\tau_j \geq 0, \;\; 1 \leq j \leq r-2 \tag{4}$$

To simplify the equations we can rewrite the negative loglikelihood as follows:

$$L = -\sum_{i=1}^{n}\sum_{k=0}^{r-1} \ln(A_{ik} - B_{ik})I(y^i = k) \tag{5}$$

where

$$A_{ik} = \begin{cases} F(\kappa_0 + \sum_{j=1}^{k} \tau_j - \xi_i), & \text{if } k = 0, \dots, r-2 \\ 1, & \text{if } k = r-1 \end{cases}$$

$$B_{ik} = \begin{cases} 0, & \text{if } k = 0 \\ F(\kappa_0 + \sum_{j=1}^{k-1} \tau_j - \xi_i), & \text{if } k = 1, \dots, r-1 \end{cases}$$

Let's introduce $L_{ik} = -\ln(A_{ik} - B_{ik})I(y^i = k)$ and then the derivative of L_{ik} with respect to κ_0 will be:

$$\begin{aligned} \frac{\partial L_{ik}}{\partial \kappa_0} &= \frac{-[A_{ik}(1-A_{ik}) - B_{ik}(1-B_{ik})]}{A_{ik} - B_{ik}} I(y^i = k) \\ &= (A_{ik} + B_{ik} - 1)I(y^i = k) \end{aligned}$$

For $j = y^i$:

$$\frac{\partial L_{ik}}{\partial \tau_j} = \frac{-A_{ik}(1-A_{ik})}{A_{ik} - B_{ik}} I(y^i = k)$$

For all $j < y^i$:

$$\frac{\partial L_{ik}}{\partial \tau_j} = (A_{ik} + B_{ik} - 1)I(y^i = k)$$

For all $j > y^i$: $\frac{\partial L_{ik}}{\partial \tau_j} = 0$.

The derivative with respect to the score ξ_i is:

$$\frac{\partial L_{ik}}{\partial \xi_i} = (-A_{ik} - B_{ik} + 1)I(y^i = k) \tag{6}$$

2.2.1 Matrix-Space Word Model

Here we show the derivatives with respect to a word. For the OLogReg model with matrix-space word representations, we have:

$$\frac{\partial L}{\partial W_{x_j^i}} = \frac{\partial L}{\partial \xi_i} \cdot \frac{\partial \xi_i}{\partial W_{x_j^i}}$$

The expression for $\frac{\partial L}{\partial \xi_i}$ is given in (6); we will derive $\frac{\partial \xi_i}{\partial W_{x_j^i}}$ from (2). In the case of the Matrix-Space word model each word is represented as an $m \times m$ affine matrix W:

$$W = \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} \tag{7}$$

We choose the class of **affine matrices** since for affine matrices matrix multiplication represents both operations: linear transformation and translation. Linear transformation is important for modeling changes in sentiment - translation is also useful (we make use of a translation vector during initialization, see Section 2.4). In this work we consider $m \geq 3$ since we want the matrix A from (7) to represent rotation and scaling. Applying the affine transformation W to vector $[x, 1]^T$ is equivalent to applying linear transformation A and translation b to x. [3]

Though vectors u and v can be learned together with word matrices W_j, **we choose to fix u and v.** The main intuition behind fixing u and v is *to reduce the degrees of freedom of the model*: different assignments of u, v and W_j-s can lead to the same score ξ, i.e. there exist $\hat{u}$ $\hat{v}$ and $\hat{W}_j$-s different from u, v and W_j-s respectively, such that $\xi(u, v, W)$ would be equal to $\xi(\hat{u}, \hat{v}, \hat{W})$. [4]

[3]
$$\begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} = \begin{pmatrix} Ax + b \\ 1 \end{pmatrix}$$
where A is a linear transformation, b is a translation vector. Also the product of affine matrices is an affine matrix.

[4] The specific choice of u and v leads to an equivalent model for all $\hat{u}$ and $\hat{v}$ such that $\hat{u} = M^T u$, $\hat{v} = M^{-1} v$, where M is any invertible transformation (i.e. $\hat{u}, \hat{v}$ are derived from u,v by applying linear transformations M^T, M^{-1} respectively):

$$\begin{aligned} u^T W_1 W_2 v &= (u^T M)(M^{-1} W_1 M)(M^{-1} W_2 M)(M^{-1} v) \\ &= \hat{u}^T \hat{W}_1 \hat{W}_2 \hat{v} \end{aligned}$$

The derivative of the phrase ξ_i with respect to j-th word W_j would be (for brevity we drop the phrase index and W_j refers to $W_{x_j^i}$ and p refers to $|x_i|$):

$$\begin{aligned} \frac{\partial \xi_i}{\partial W_j} &= \left(\frac{\partial u^T W_1 W_2 \dots W_p v}{\partial W_j} \right) \\ &= \left[(u^T W_1 \dots W_{j-1})^T (W_{j+1} \dots W_p v)^T \right] \\ &= \left[(W_{j-1}^T \dots W_1^T)(u v^T)(W_p^T \dots W_{j+1}^T) \right] \end{aligned}$$

(see Peterson and Pederson(2008)).
In case if a certain word appears multiple times in the phrase, the derivative with respect to that word would be a sum of derivatives with respect to each appearance of a word, while all other appearances are fixed. For example,

$$\left(\frac{\partial u^T W W_1 W v}{\partial W} \right) = u(W_1 W v)^T + (u^T W W_1)^T v^T$$

where W is a representation of a word that is repeated.

So given the expression (6) for $\frac{\partial L}{\partial \xi_i}$, the derivative with respect to each word can be computed. Notice that the update for the j-th word in a sentence depends on the order words, which is in line with our desire to account for word order.

2.2.2 Optimization

The goal of training procedure is for the i-th phrase with p words $x_1 x_2 \dots x_p$ to learn word matrices W_1, W_2, ..., W_p such that resulting ξ_i-s will lead to the lowest negative loglikelihood. The goal of training procedure is to find word matrices W_1, W_2, ... W_p and thresholds κ_0, τ_1, ... τ_{r-2} such that the negative loglikelihood is minimized. So, given the negative loglikelihood and the derivatives with respect κ_0 and τ_j-s and word matrices W, we optimize objective (5) subject to $\tau_j \geq 0$. We use L-BFGS-B (Large-scale Bound-constrained Optimization) by Byrd et al. (1995) as an optimizer.

2.2.3 Regularization in Matrix-Space Model

In order to make sure that the L-BFGS-B updates do not cause numerical issues we perform the following regularization to the resulting matrices. An m by m matrix W_j that can be represented as:

$$W_j = \begin{pmatrix} A_{11} & a_{12} \\ a_{21}^T & a_{22} \end{pmatrix}$$

where $A_{11} \in \mathbb{R}^{m-1 \times m-1}$, $a_{12}, a_{21} \in \mathbb{R}^{m-1 \times 1}$, $a_{22} \in \mathbb{R}$. First make the matrix affine by updating the last row, then the updated matrix will look like:

$$\hat{W}_j = \begin{pmatrix} A_{11} & a_{12} \\ 0 & 1 \end{pmatrix}$$

It can be proven that such a projection returns the closest affine matrix in Frobenius norm.

However, we also want to regularize the model to avoid ill-conditioned matrices. Ill-conditioned matrices represent transformations whose output is very sensitive to small changes in the input and therefore they have a similar effect to having large weights in a bag-of-words model. To perform such a regularization we "shrink" the singular values of A_{11} towards one. More specifically, we first use the Singular Value Decomposition (SVD) of the A_{11}: $U\Sigma V^T = A_{11}$, where U and V are orthogonal matrices, Σ is a matrix with singular values on the diagonal. Then we update singular values in the following way to get $\tilde{\Sigma}$: $\tilde{\Sigma}_{ii} = \Sigma_{ii}^h$, where h is a parameter between 0 and 1. If $h = 1$ then Σ_{ii} remains the same. In the extreme case $h = 0$ then $\Sigma_{ii}^h = 1$. For intermediate values of h the singular values of A_{11} would be brought closer to one. Finally, we recompute $\tilde{A}_{11}$: $\tilde{A}_{11} = U\tilde{\Sigma}V^T$. So, W_j would be :

$$\tilde{W}_j = \begin{pmatrix} \tilde{A}_{11} & a_{12} \\ 0 & 1 \end{pmatrix}$$

2.2.4 Learning in the Matrix-Space Model

We use Algorithm 1 to learn the matrix-space model. What essentially happens is that we iterate two steps: optimizing the W matrices using L-BFGS-B and the projection step. L-BFGS-B returns a solution that is not necessarily an affine matrix. After projecting to the space of affine matrices we start L-BFGS-B from a better initial point. In practice, the first few iterations lead to large decrease in negative loglikelihood.

2.3 Bag-Of-Words Model

In the bag-of-words model the score of the i-th phrase is given in (1). Therefore, the partial derivative with respect to j-th word in i-th phrase $\frac{\partial \xi_i}{\partial w_{x_j^i}}$ is equal to the number c_j of times x_i^j appears in x^i, so:

$$\frac{\partial L}{\partial w_{x_j^i}} = \frac{\partial L}{\partial \xi_i} \cdot c_j$$

Algorithm 1 Training Algorithm for Matrix-Space OLogReg

```
Input: {(x_1, y_1), ..., (x_n, y_n)} //training data
Input: h //projection parameter
Input: T //number of iterations
Input: W, κ_0 and τ_j //initial values
for t = 1, ..., T do
    (W, κ_0, τ_j)=minimize L using L-BFGS-B
    for i = 1, ..., d do
        W_i=Project(W_i, h)
    end for
end for
Return W, κ_0, τ_j
```

Optimization. We minimize negative loglikelihood using L-BFGS-B subject to $\tau_j \geq 0$.

Regularization. To prevent overfitting for bag-of-words model we regularize w. The L_2-regularized negative loglikelihood will consist of the expression in (5) and an additional term $\frac{\lambda}{2}||w||_2^2$, where $||\cdot||_2$ is the L_2-norm of a vector. The derivative of the additional term with respect to w will be:

$$\frac{\partial \frac{\lambda}{2}||w||_2^2}{\partial w} = \lambda w$$

Hence the partial derivative with respect to $w_{x_j^i}$ will have an additional term $\lambda w_{x_j^i}$.

2.4 Initialization

Initialization of bag-of-words OLogReg. We initialize the weight for each word with zero and κ_0 with a random number and τ_j-s with non-negative random numbers. Since the learning problem for bag-of-words OLogReg is convex, we will get the global optimum.

Better Initialization of Matrix-Space Model. Preliminary experiments showed that the Matrix-Space model needs a good initialization. Initializing with different random matrices reaches different local minima and the quality of local minima depends on initialization. Therefore, it is important to initialize the model with a good initial point. *One way to initialize the Matrix-Space model is to use the weights learned by the bag-of-words model.* We use the following intuition for initializing the Matrix-Space model. As noted in Section 2.2.1 applying transformation A of affine matrix W can model a linear

transformation, while vector b represents a translation. Since matrix-space model can encode a vector-space model (Rudolph and Giesbrecht, 2010), we can initialize the matrices to exactly mimic the bag-of-words model. In order to do that we place the weight, learned by the bag-of-words model in the first component of b. Let's assume that w_{x_1} and w_{x_2} are the weights learned for two distinct words x_1 and x_2 respectively. To compute the polarity score of a phrase x_1, x_2 the bag-of-words model sums the weights of these two words: w_{x_1} and w_{x_2}. Now we want to have the same effect in matrix-space model. Here we assume $m = 3$.

$$Z = \begin{pmatrix} 1 & 0 & w_{x_1} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & w_{x_2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & w_{x_1} + w_{x_2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Finally, there is a step of mapping matrix Z to a number using u and v, such that $\xi(Z) = w_{x_1} + w_{x_2}$.We also want vector u and v to be such that:

$$u^T \begin{pmatrix} 1 & 0 & w_{x_1} + w_{x_2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} v = w_{x_1} + w_{x_2} \quad (8)$$

The last equation can help us construct u and v. We also set u and v to be orthogonal: $u^T v = 0$. So, we arbitrarily choose two orthogonal vectors for which equation (8) holds: $u = [1, \sqrt{2}, 1]^T$ and $v = [1, -\sqrt{2}, 1]^T$.[5]

3 Experimental Methodology

For experimental evaluation of the proposed method we use the publicly available Multi-Perspective Question Answering (MPQA)[6] corpus (Wiebe et al., 2005) version 1.2, which contains 535 newswire documents that are manually annotated with phrase-level subjectivity and intensity. We use the expression-level boundary markings in MPQA to extract phrases. We evaluate on positive, negative and neutral opinion expressions that have intensities "medium", "high" or "extreme".[7] The schematic mapping of phrase polarity and intensity values on ordinal sentimental scale is shown in Table 1.

Polarity	Intensity	Ordinal label
negative	high, extreme	0
negative	medium	1
neutral	high, extreme, medium	2
positive	medium	3
positive	high, extreme	4

Table 1: Mapping of combination of polarities and intensities from MPQA dataset to our ordinal sentiment scale.

3.1 Training Details

We perform 10-fold cross-validation on phrases extracted from the MPQA corpus: eight folds for training; one as a validation set; and one as test set. In total there were 8022 phrases. Before training, we extract lemmas for each word. For evaluation we use Ranking Loss: $\frac{1}{n}\sum_i |\hat{y}^i - y^i|$, where $\hat{y^i}$ is the prediction.

Choice of dimensionality m. The reported experiments are done by setting $m = 3$. Preliminary experiments with higher values of m (5, 20, 50), did not lead to a better performance and increased the training time; therefore we did not use those values in our final experiments.

3.2 Methods

PRank. For each of the folds, we run 500 iterations of PRank and choose an early stopping iteration using a model that led to the lowest ranking loss on the validation set; afterwards report the average performance of on a test set.

Bag-of-words OLogReg. To prevent overfitting we search for the best regularization parameter among the following values of λ: 10^i, from 10^{-4} to 10^4. The lowest negative log-likelihood value on the validation set is attained for[8] $\lambda = 0.1$. With this value of λ fixed, the final model is the one with the lowest negative loglikelihood on the training set.

[5] If $m > 3$, u and v can be set using the same intuition.

[6] http://www.cs.pitt.edu/mpqa/

[7] We ignored low-intensity phrases similar to (Choi and Cardie, 2008; Nakagawa et al., 2010).

[8] We pick single λ that gives best average validation set performance, and then use it to compute the average test set performance.

Method	Ranking loss
PRank	0.7808
Bag-of-words OLogReg	0.6665
Matrix-space OLogReg+RandInit	0.7417
Matrix-space OLogReg+BowInit	0.6375†

Table 2: Ranking loss for vector-space Ordered Logistic Regression and Matrix-Space Logistic Regression.
† Stands for a significant difference w.r.t. the Bag-Of-Words OLogReg model with p-value less than 0.001 ($p < 0.001$)

Matrix-space OLogReg+RandInit. First, we initialized matrices with with random numbers from normal distribution $N(0, 0.1)$ and set u and v as in section 2.4, T is set to 25. We run with two different random seeds and three different values for the parameter h: [0.1, 0.5, 0.9] and report the performance of the model that had the lowest likelihood on the validation set. The setting of h that lead to the best model was 0.9.

Matrix-space OLogReg+BowInit. For the matrix-space models we initialize the model with the output of the regularized Bag-of-words OLogReg as described in Section 2.4, T is set to 25. Then we use the training procedure of Algorithm 1. We consider three different values for the parameter h [0.1, 0.5, 0.9] and choose as the model with the lowest validation set negative log-likelihood. The best setting of h was 0.1.

4 Results and Discussion

We report Ranking Loss for the four models in Table 2. The worst performance (denoted by the highest ranking loss value) is obtained by PRank, followed by matrix-space OLogReg with random initialization. Bag-of-words OLogReg obtains quite good performance, and matrix-space OLogReg, initialized using the bag-of-words model performs the best, showing statistically significant improvements over the bag-of-words OLogReg model according to a paired t-test. .

To see what the bag-of-word and matrix-space models are learning we performed inference on a few examples. In Table 3 we show the sentiment scores of the best performing bag-of-words **OLogReg** model and the best performing model based on matrices **Matrix-space OLogReg+BowInit**. By sentiment score, we mean equation (1) of Bag-of-words OLogReg and equation (2) of Matrix-space OLogReg+BowInit.

Phrase	Matrix-space OLogReg+BowInit	Bag-of-words OLogReg
not	-0.83	-0.42
very	0.23	0.04
good	2.81	1.51
very good	3.53	1.55
not good	-0.16	1.09
not very good	0.66	1.13
bad	-1.67	-1.42
very bad	-2.01	-1.38
not bad	-0.54	-1.85
not very bad	-1.36	-1.80

Table 3: Phrase and the sentiment scores of the phrase for 2 models Matrix-space OLogReg+BowInit and Bag-of-words OLogReg respectively. Notice that **relative ranking order what matters**

Here we choose two popular adjectives like 'good' and 'bad' that appeared in the training data, and examine the effect of applying the intensifier 'very' on the sentiment score. As we can see, the matrix-space model learns a matrix that intensifies both 'bad' and 'good' in the correct sentiment scale, i.e., $\xi(good) < \xi(very\ good)$ and $\xi(bad) < \xi(very\ bad)$, while the bag-of-words model gets the sentiment of 'very bad' wrong: it is more positive than 'bad'. We also looked at the effect of combining 'not' with these adjectives. The matrix-space model correctly encodes the effect of the negator for both positive and negative adjectives, such that $\xi(not\ good) < \xi(good)$ and $\xi(bad) < \xi(not\ bad)$. For the interesting case of applying a negator to a phrase with an intensifier, $\xi(not\ good)$ should be less than $\xi(not\ very\ good)$ and $\xi(not\ very\ bad)$ should be less than $\xi(not\ bad)$.[9] As shown in Table 3, these are predicted correctly by the matrix-space model, which the matrix-space model gets right, but the bag-of-words model misses in the case of "bad".

Also notice that since in the matrix-space model

[9] See the detailed discussion in Taboada et al. (2011) and Liu and Seneff (2009).

each word is represented as a function, more specifically a linear operator, and the function composition defined as matrix multiplication, we can think of "not very" being an operator itself, that is a composition of operator "not" and operator "very".

5 Related Work

Sentiment Analysis. There has been a lot of research in determining the sentiment of words and constructing polarity dictionaries (Hatzivassiloglou and McKeown, 1997; Wiebe, 2000; Rao and Ravichandran, 2009; Mohammad et al., 2009; Velikovich et al., 2010). Some recent work is trying to identify the degree of sentiment of adjectives and adverbs from text using co-occurrence statistics. Work by Taboada et. al (2011) and Liu and Seneff (2009), suggest ways of computing the sentiment of adjectives from data, and computing the effect of combining adjective with adverb as multiplicative effect and combining adjective with negation as additive effect. However these models require the knowledge of a part of speech of given words and the list of negators (since the negator is an adjective as well). In our work we propose a single unified model for handling all words of any part of speech.

On the other hand, there has been some research in trying to model compositional effects for sentiment at the phrase- and sentence-level. Choi and Cardie (2008) hand-code compositional rules in order to model compositional effects of combining different words in the phrase. The hand-coded rules are based on domain knowledge and used to *learn* the effects of combining words in the phrase. Another recent work that tries to model the compositional semantics of combining different words is Nakagawa et. al. (2010), which proposes a model that learns the effects of combining different words using phrase/sentence dependency parse trees and an initial polarity dictionary. They present a learning method that employs hidden variables for sentiment classification: given the polarity of a sentence and the *a priori* polarities of its words, they learn how to model the interactions between words with head-modifier relations in the dependency tree.

Some of the previous work looked at MPQA phrase-level classification. Wilson et al. (2004) tackles the problem of classifying clauses according to their subjective strength but not polarity; Wilson et al. (2005) classifies phrases according to their polarity/sentiment but not strength. Our task is different: we classify phrases according to a single ordinal scale that combines both polarity and strength.

Task of predicting document-level star ratings was considered in (Pang and Lee, 2005; Goldberg and Zhu, 2006). In the current work we look at fine-grained sentiment analysis, more specifically we study word representations for use in true compositional semantic settings.

Distributional Semantics and Compositionality. Research in the area of distributional semantics in NLP and Cognitive Science has looked at different word representations and different ways of combining words. Mitchell and Lapata (2010) propose a framework for vector-based semantic composition. They define composition as an additive or multiplicative function of two vectors and show that compositional approaches generally outperform non-compositional approaches that treat the phrase as the union of single lexical items.

Work by Baroni and Zamparelli (2010) models nouns as vectors in some semantic space and adjectives as matrices. It shows that modeling adjectives as linear transformations and applying those linear transformations to nouns results in final vectors for adjective-noun compositions that are close in semantic space to other similar phrases. The authors argue that modeling adjectives as a linear transformation is a better idea than using additive vector-space models. In this work, a separate matrix for each adjective is *learned* using the Partial Least Squares method in a completely unsupervised way. The recent paper by Rudolph and Giesbrecht (2010), described in the introduction, argues for *multiplicative matrix-space* models. In contrast to other work in this area, our work is concerned with a specific dimension of word meaning — sentiment. Our techniques, however, are quite general and should be applicable to other problems in lexical semantics.

6 Conclusions and Future work

In the current work we present a novel matrix-space model for ordinal scale sentiment prediction and an algorithm for learning such a model. The proposed

model learns a matrix for each word; the composition of words is modeled as iterated matrix multiplication. The matrix-space framework with iterated matrix multiplication defines an elegant framework for modeling composition; it is also quite general. We use the matrix-space framework in the context of sentiment prediction, a domain where interesting compositional effects can be observed. The main focus of this work was to study word representations (represent as a single weight vs. as a matrix) for use in true compositional semantic settings. One of the benefits of the proposed approach is that by learning matrices for words, the model can handle unseen word compositions (e.g. unseen bigrams) when the unigrams involved have been seen.

However, it is not trivial to learn a matrix-space model. Since the final optimization problem is non-convex, the initialization has to be done carefully. Here the weights learned in bag-of-words model come to rescue and provide good initial point for optimization procedure. The final model outperforms the bag-of-words based model, which suggests that this research direction is very promising.

Though in our model the order of composition is the same as the word order, we believe that a linguistically informed order of composition can give us further performance gains. For example, one can use the output of a dependency parser to guide the order of composition, similar to Nakagawa et al. (2010). Another possibility for improvement is to use the information about the scope of negation. In the current work we assume the scope of negation to be the expression following the negation; in reality, however, determining the scope of negation is a complex linguistic phenomenon (Moilanen and Pulman, 2007). So the proposed model can benefit from identifying the scope of negation, similar to (Councill et al., 2010).

Also we plan to consider other ways to initialize the matrix-space model. One interesting direction to explore might be to use non-negative matrix factorization (Lee and Seung, 2001), co-clustering techniques (Dhillon, 2001) to better initialize words that share similar contexts. The other possible direction is to use existing sentiment lexicons and employing a "curriculum learning" strategy (Bengio et al., 2009; Kumar et al., 2010) for our learning problem.

Acknowledgments

This work was supported in part by National Science Foundation Grants BCS-0904822, BCS-0624277, IIS-0968450; and by a gift from Google. We thank the anonymous reviewers, and David Bindel, Nikos Karampatziakis, Lillian Lee and Cornell NLP group for useful suggestions and insightful discussions.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1183–1193, Morristown, NJ, USA. Association for Computational Linguistics.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09. ACM.

R. H. Byrd, P. Lu, and J. Nocedal. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, pages 1190–1208.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Isaac G. Councill, Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, NeSp-NLP '10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koby Crammer and Yoram Singer. 2001. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2010. Was it good? It was provocative. learning the meaning of scalar adjectives. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, July 11–16. ACL.

I. S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*.

Andrew B. Goldberg and Jerry Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*.

James W. Hardin and Joseph Hilbe. 2007. *Generalized Linear Models and Extensions*. Stata Press.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *EACL*, pages 174–181.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2, Special Issue on Sentiment Analysis)):110–125.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems 23*. NIPS.

D. Lee and H. Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*.

Jingjing Liu and Stephanie Seneff. 2009. Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 161–169, Singapore, August. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Saif Mohammad, Cody Dunne, and Bonnie Dorr. 2009. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 599–608, Singapore, August. Association for Computational Linguistics.

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007)*, pages 378–382, September 27-29.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

K. B. Petersen and M. S. Pedersen. "2008". *The Matrix Cookbook*. "Technical University of Denmark", "oct". "Version 20081110".

Livia Polanyi and Annie Zaenen. 2004. Contextual lexical valence shifters. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 675–682, Athens, Greece, March. Association for Computational Linguistics.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 907–916, Morristown, NJ, USA. Association for Computational Linguistics.

Mostafa Shaikh, Helmut Prendinger, and Ishizuka Mitsuru. 2007. Assessing sentiment of text by semantic dependency and contextual valence analysis.

Maite Taboada, Julian Brooke, Milan Tofiloskiy, and Kimberly Vollz. 2011). Lexicon-based methods for sentiment analysis. In *Computational Linguistics*.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 777–785, Los Angeles, California, June. Association for Computational Linguistics.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation (formerly Computers and the Humanities)*, 39(2/3):164–210.

Janyce M. Wiebe. 2000. Learning subjective adjectives from corpora. In *In AAAI*, pages 735–740.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? In *AAAI*. AAAI.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Training a Parser for Machine Translation Reordering

Jason Katz-Brown Slav Petrov Ryan McDonald Franz Och
David Talbot Hiroshi Ichikawa Masakazu Seno Hideto Kazawa
Google
{jasonkb|slav|ryanmcd|och|talbot|ichikawa|seno|kazawa}@google.com

Abstract

We propose a simple training regime that can improve the extrinsic performance of a parser, given only a corpus of sentences and a way to automatically evaluate the extrinsic quality of a candidate parse. We apply our method to train parsers that excel when used as part of a reordering component in a statistical machine translation system. We use a corpus of weakly-labeled reference reorderings to guide parser training. Our best parsers contribute significant improvements in subjective translation quality while their intrinsic attachment scores typically regress.

1 Introduction

The field of syntactic parsing has received a great deal of attention and progress since the creation of the Penn Treebank (Marcus et al., 1993; Collins, 1997; Charniak, 2000; McDonald et al., 2005; Petrov et al., 2006; Nivre, 2008). A common—and valid—criticism, however, is that parsers typically get evaluated only on Section 23 of the Wall Street Journal portion of the Penn Treebank. This is problematic for many reasons. As previously observed, this test set comes from a very narrow domain that does not necessarily reflect parser performance on text coming from more varied domains (Gildea, 2001), especially web text (Foster, 2010). There is also evidence that after so much repeated testing, parsers are indirectly over-fitting to this set (Petrov and Klein, 2007). Furthermore, parsing was never meant as a stand-alone task, but is rather a means to an end, towards the goal of building systems that can process natural language input.

This is not to say that parsers are not used in larger systems. All to the contrary, as parsing technology has become more mature, parsers have become efficient and accurate enough to be useful in many natural language processing systems, most notably in machine translation (Yamada and Knight, 2001; Galley et al., 2004; Xu et al., 2009). While it has been repeatedly shown that using a parser can bring net gains on downstream application quality, it is often unclear how much intrinsic parsing accuracy actually matters.

In this paper we try to shed some light on this issue by comparing different parsers in the context of machine translation (MT). We present experiments on translation from English to three Subject-Object-Verb (SOV) languages,[1] because those require extensive syntactic reordering to produce grammatical translations. We evaluate parse quality on a number of extrinsic metrics, including word reordering accuracy, BLEU score and a human evaluation of final translation quality. We show that while there is a good correlation between those extrinsic metrics, parsing quality as measured on the Penn Treebank is not a good indicator of the final downstream application quality. Since the word reordering metric can be computed efficiently offline (i.e. without the use of the final MT system), we then propose to tune parsers specifically for that metric, with the goal of improving the performance of the overall system.

To this end we propose a simple training regime

[1] We experiment with Japanese, Korean and Turkish, but there is nothing language specific in our approach.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 183–192,
Edinburgh, Scotland, UK, July 27–31, 2011.

which we refer to as *targeted self-training* (Section 2). Similar to self-training, a baseline model is used to produce predictions on an unlabeled data set. However, rather than directly training on the output of the baseline model, we generate a list of hypotheses and use an external signal to select the best candidate. The selected parse trees are added to the training data and the model is then retrained. The experiments in Section 5 show that this simple procedure noticeably improves our parsers for the task at hand, resulting in significant improvements in downstream translation quality, as measured in a human evaluation on web text.

This idea is similar in vein to McClosky. et al. (2006) and Petrov et al. (2010), except that we use an extrinsic quality metric instead of a second parsing model for making the selection. It is also similar to Burkett and Klein (2008) and Burkett et al. (2010), but again avoiding the added complexity introduced by the use of additional (bilingual) models for candidate selection.

It should be noted that our extrinsic metric is computed from data that has been manually annotated with reference word reorderings. Details of the reordering metric and the annotated data we used are given in Sections 3 and 4. While this annotation requires some effort, such annotations are much easier to obtain than full parse trees. In our experiments in Section 6 we show that we can obtain similar improvements on downstream translation quality by targeted self-training with weakly labeled data (in form of word reorderings), as with training on the fully labeled data (with full syntactic parse trees).

2 Targeted Self-Training

Our technique for retraining a baseline parser is an extension of self-training. In standard parser self-training, one uses the baseline parsing model to parse a corpus of sentences, and then adds the 1-best output of the baseline parser to the training data. To target the self-training, we introduce an additional step, given as Algorithm 1. Instead of taking the 1-best parse, we produce a ranked n-best list of predictions and select the parser which gives the best score according to an external evaluation function. That is, instead of relying on the intrinsic model score, we use an extrinsic score to select the parse towards which to update. In the case of a tie, we prefer the parse ranked most highly in the n-best list.

Algorithm 1 Select parse that maximizes an extrinsic metric.

Input: baseline parser B
Input: sentence S
Input: function COMPUTEEXTRINSIC(parse P)
Output: a parse for the input sentence

```
P^n = {P_1, ..., P_n}   ← n-best parses of S by B
maxScore = 0
bestParse = ∅
for k = 1 to n do
    extrinsicScore = COMPUTEEXTRINSIC(P_k)
    if extrinsicScore > maxScore then
        maxScore = extrinsicScore
        bestParse = P_k
    end if
end for
return bestParse
```

The motivation of this selection step is that good performance on the downstream external task, measured by the extrinsic metric, should be predictive of an intrinsically good parse. At the very least, even if the selected parse is not syntactically correct, or even if it goes against the original treebanking guidelines, it results in a higher extrinsic score and should therefore be preferred.

One could imagine extending this framework by repeatedly running self-training on successively improving parsers in an EM-style algorithm. A recent work by Hall et al. (2011) on training a parser with multiple objective functions investigates a similar idea in the context of online learning.

In this paper we focus our attention on machine translation as the final application, but one could envision applying our techniques to other applications such as information extraction or question answering. In particular, we explore one application of targeted self-training, where computing the extrinsic metric involves plugging the parse into an MT system's reordering component and computing the accuracy of the reordering compared to a reference word order. We now direct our attention to the details of this application.

3 The MT Reordering Task

Determining appropriate target language word order for a translation is a fundamental problem in MT. When translating between languages with significantly different word order such as English and Japanese, it has been shown that metrics which explicitly account for word-order are much better correlated with human judgments of translation quality than those that give more weight to word choice, like BLEU (Lavie and Denkowski, 2009; Isozaki et al., 2010a; Birch and Osborne, 2010). This demonstrates the importance of getting reordering right.

3.1 Reordering as a separately evaluable component

One way to break down the problem of translating between languages with different word order is to handle reordering and translation separately: first reorder source-language sentences into target-language word order in a preprocessing step, and then translate the reordered sentences. It has been shown that good results can be achieved by reordering each input sentence using a series of tree transformations on its parse tree. The rules for tree transformation can be manually written (Collins et al., 2005; Wang, 2007; Xu et al., 2009) or automatically learned (Xia and McCord, 2004; Habash, 2007; Genzel, 2010).

Doing reordering as a preprocessing step, separately from translation, makes it easy to evaluate reordering performance independently from the MT system. Accordingly, Talbot et al. (2011) present a framework for evaluating the quality of reordering separately from the lexical choice involved in translation. They propose a simple *reordering metric* based on METEOR's reordering penalty (Lavie and Denkowski, 2009). This metric is computed solely on the source language side. To compute it, one takes the candidate reordering of the input sentence and partitions it into a set $\mathcal{C}$ of contiguous spans whose content appears contiguously in the same order in the reference. The reordering score is then computed as

$$\rho(\mathbf{e}_{\text{sys}}, \mathbf{e}_{\text{ref}}) = 1 - \frac{|\mathcal{C}| - 1}{|\mathbf{e}| - 1}.$$

This metric assigns a score between 0 and 1 where 1 indicates that the candidate reordering is identical to the reference and 0 indicates that no two words that are contiguous in the candidate reordering are contiguous in the reference. For example, if a reference reordering is `A B C D E`, candidate reordering `A B E C D` would get score $1-(3-1)/(5-1) = 0.5$.

Talbot et al. (2011) show that this reordering score is strongly correlated with human judgment of translation quality. Furthermore, they propose to evaluate the reordering quality of an MT system by computing its reordering score on a test set consisting of source language sentences and their reference reorderings. In this paper, we take the same approach for evaluation, and in addition, we use corpora of source language sentences and their reference reorderings for training the system, not just testing it. We describe in more detail how the reference reordering data was prepared in Section 4.1.

3.2 Reordering quality as predictor of parse quality

Figure 1 gives concrete examples of good and bad reorderings of an English sentence into Japanese word order. It shows that a bad parse leads to a bad reordering (lacking inversion of verb "wear" and object "sunscreen") and a low reordering score. Could we flip this causality around, and perhaps try to identify a good parse tree based on its reordering score? With the experiments in this paper, we show that indeed a high reordering score is predictive of the underlying parse tree that was used to generate the reordering being a good parse (or, at least, being good enough for our purpose).

In the case of translating English to Japanese or another SOV language, there is a large amount of reordering required, but with a relatively small number of reordering rules one can cover a large proportion of reordering phenomena. Isozaki et al. (2010b), for instance, were able to get impressive English→Japanese results with only a single reordering rule, given a suitable definition of a *head*. Hence, the reordering task depends crucially on a correct syntactic analysis and is extremely sensitive to parser errors.

4 Experimental Setup

4.1 Treebank data

In our experiments the baseline training corpus is the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al., 1993) using standard training/development/testing splits. We converted the treebank to match the tokenization expected by our MT system. In particular, we split tokens containing hyphens into multiple tokens and, somewhat simplistically, gave the original token's part-of-speech tag to all newly created tokens. In Section 6 we make also use of the Question Treebank (QTB) (Judge et al., 2006), as a source of syntactically annotated out-of-domain data. Though we experiment with both dependency parsers and phrase structure parsers, our MT system assumes dependency parses as input. We use the Stanford converter (de Marneffe et al., 2006) to convert phrase structure parse trees to dependency parse trees (for both treebank trees and predicted trees).

4.2 Reference reordering data

We aim to build an MT system that can accurately translate typical English text that one finds on the Internet to SOV langauges. To this end, we randomly sampled 13595 English sentences from the web and created Japanese-word-order reference reorderings for them. We split the sentences arbitrarily into a 6268-sentence Web-Train corpus and a 7327-sentence Web-Test corpus.

To make the reference alignments we used the technique suggested by Talbot et al. (2011): ask annotators to translate each English sentence to Japanese extremely literally and annotate which English words align to which Japanese words. Golden reference reorderings can be made programmatically from these annotations. Creating a large set of reference reorderings is straightforward because annotators need little special background or training, as long as they can speak both the source and target languages. We chose Japanese as the target language through which to create the English reference reorderings because we had access to bilingual annotators fluent in English and Japanese.

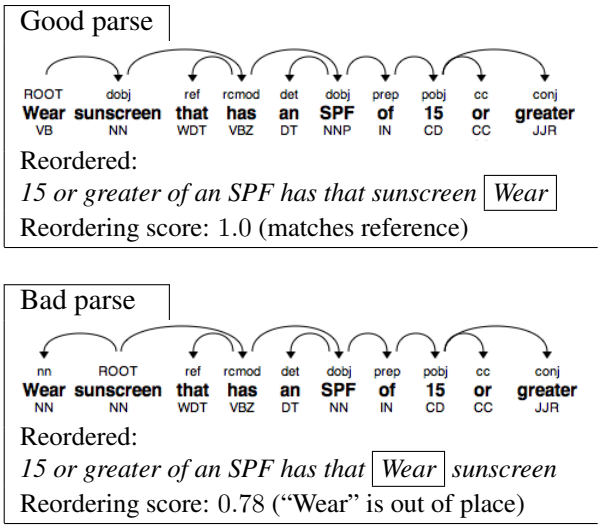


Figure 1: Examples of good and bad parses and corresponding reorderings for translation from English to Japanese. The good parse correctly identifies "Wear" as the main verb and moves it to the end of the sentence; the bad parse analyses "Wear sunscreen" as a noun phrase and does not reorder it. This example was one of the wins in the human evaluation of Section 5.2.

4.3 Parsers

The core dependency parser we use is an implementation of a transition-based dependency parser using an arc-eager transition strategy (Nivre, 2008). The parser is trained using the averaged perceptron algorithm with an early update strategy as described in Zhang and Clark (2008). The parser uses the following features: word identity of the first two words on the buffer, the top word on the stack and the head of the top word on the stack (if available); part-of-speech identities of the first four words on the buffer and top two words on the stack; dependency arc label identities for the top word on the stack, the left and rightmost modifier of the top word on the stack, and the leftmost modifier of the first word in the buffer. We also include conjunctions over all non-lexical features.

We also give results for the latent variable parser (a.k.a. BerkeleyParser) of Petrov et al. (2006). We convert the constituency trees output by the BerkeleyParser to labeled dependency trees using the same procedure that is applied to the treebanks.

While the BerkeleyParser views part-of-speech (POS) tagging as an integral part of parsing, our dependency parser requires the input to be tagged

with a separate POS tagger. We use the TnT tagger (Brants, 2000) in our experiments, because of its efficiency and ease of use. Tagger and parser are always trained on the same data.

For all parsers, we lowercase the input at train and test time. We found that this improves performance in parsing web text. In addition to general uppercase/lowercase noisiness of the web text negatively impacting scores, we found that the baseline case-sensitive parsers are especially bad at parsing imperative sentences, as discussed in Section 5.3.2.

4.4 Reordering rules

In this paper we focus on English to Japanese, Korean, and Turkish translation. We use a superset of the reordering rules proposed by Xu et al. (2009), which flatten a dependency tree into SOV word order that is suitable for all three languages. The rules define a precedence order for the dependents of each part of speech. For example, a slightly simplified version of the precedence order of child labels for a verbal head HEADVERB is: advcl, nsubj, prep, [other children], dobj, prt, aux, neg, HEADVERB, mark, ref, compl.

Alternatively, we could have used an automatic reordering-rule learning framework like that of Genzel (2010). Because the reordering accuracy metric can be computed for any source/target language pair, this would have made our approach language completely independent and applicable to any language pair. We chose to use manually written rules to eliminate the variance induced by the automatic reordering-rule learning framework.

4.5 MT system

We carried out all our translation experiments on a state-of-the-art phrase-based statistical MT system. During both training and testing, the system reorders source-language sentences in a preprocessing step using the above-mentioned rules. During decoding, we used an allowed jump width of 4 words. In addition to the regular distance distortion model, we incorporate a maximum entropy based lexicalized phrase reordering model (Zens and Ney, 2006) as a feature used in decoding.

Overall for decoding, we use between 20 to 30 features, whose weights are optimized using MERT (Och, 2003). All experiments for a given language pair use the same set of MERT weights tuned on a system using a separate parser (that is neither the baseline nor the experiment parser). This potentially underestimates the improvements that can be obtained, but also eliminates MERT as a possible source of improvement, allowing us to trace back improvements in translation quality directly to parser changes.[2]

For parallel training data, we use a custom collection of parallel documents. They come from various sources with a substantial portion coming from the web after using simple heuristics to identify potential document pairs. For all language pairs, we trained on approximately 300 million source words each.

5 Experiments Reordering Web Text

We experimented with parsers trained in three different ways:

1. Baseline: trained only on WSJ-Train.
2. Standard self-training: trained on WSJ-Train and 1-best parse of the Web-Train set by baseline parser.
3. Targeted self-training: trained on WSJ-Train and, for each sentence in Web-Train, the parse from the baseline parser's 512-best list that when reordered gives the highest reordering score.[3]

5.1 Standard self-training vs targeted self-training

Table 1 shows that targeted self-training on Web-Train significantly improves Web-Test reordering score more than standard self-training for both the shift-reduce parser and for the BerkeleyParser. The reordering score is generally divorced from the attachment scores measured on the WSJ-Test treebank: for the shift-reduce parser, Web-Test reordering score and WSJ-Test labeled attachment score

[2] We also ran MERT on all systems and the pattern of improvement is consistent, but sometimes the improvement is bigger or smaller after MERT. For instance, the BLEU delta for Japanese is +0.0030 with MERT on both sides as opposed to +0.0025 with no MERT.

[3] We saw consistent but diminishing improvements as we increased the size of the n-best list.

Parser	Web-Test reordering	WSJ-Test LAS
Shift-reduce WSJ baseline	0.757	85.31%
+ self-training 1x	0.760	85.26%
+ self-training 10x	0.756	84.14%
+ targeted self-training 1x	**0.770**	85.19%
+ targeted self-training 10x	**0.777**	84.48%
Berkeley WSJ baseline	0.780	88.66%
+ self-training 1x	0.785	89.21%
+ targeted self-training 1x	**0.790**	89.32%

Table 1: English→Japanese reordering scores on Web-Test for standard self-training and targeted self-training on Web-Train. Label "10x" indicates that the self-training data was weighted 10x relative to the WSJ training data. Bolded reordering scores are different from WSJ-only baseline with 95% confidence but are not significantly different from each other within the same group.

English to	BLEU		Human evaluation (scores range 0 to 6)		
	WSJ-only	Targeted	WSJ-only	Targeted	Sig. difference?
Japanese	0.1777	0.1802	2.56	2.69	yes (at 95% level)
Korean	0.3229	0.3259	2.61	2.70	yes (at 90% level)
Turkish	0.1344	0.1370	2.10	2.20	yes (at 95% level)

Table 2: BLEU scores and human evaluation results for translation between three language pairs, varying only the parser between systems. "WSJ-only" corresponds to the baseline WSJ-only shift-reduce parser; "Targeted" corresponds to the Web-Train targeted self-training 10x shift-reduce parser.

(LAS) are anti-correlated, but for BerkeleyParser they are correlated. Interestingly, weighting the self-training data more seems to have a negative effect on both metrics.[4]

One explanation for the drops in LAS is that some parts of the parse tree are important for downstream reordering quality while others are not (or only to a lesser extent). Some distinctions between labels become less important; for example, arcs labeled "amod" and "advmod" are transformed identically by the reordering rules. Some semantic distinctions also become less important; for example, any sane interpretation of "red hot car" would be reordered the same, that is, not at all.

5.2 Translation quality improvement

To put the improvement of the MT system in terms of BLEU score (Papineni et al., 2002), a widely used metric for automatic MT evaluation, we took 5000 sentences from Web-Test and had humans generate reference translations into Japanese, Korean, and Turkish. We then trained MT systems varying only the parser used for reordering in training and decoding. Table 2 shows that targeted self-training data increases BLEU score for translation into all three languages.

In addition to BLEU increase, a side-by-side human evaluation on 500 sentences (sampled from the 5000 used to compute BLEU scores) showed a statistically significant improvement for all three languages (see again Table 2). For each sentence, we asked annotators to simultaneously score both translations from 0 to 6, with guidelines that 6="Perfect", 4="Most Meaning/Grammar", 2="Some Meaning/Grammar", 0="Nonsense". We computed confidence intervals for the average score difference using bootstrap resampling; a difference is significant if the two-sided confidence interval does not include 0.

5.3 Analysis

As the divergence between the labeled attachment score on the WSJ-Test data and the reordering score on the WSJ-Test data indicates, parsing web text

[4]We did not attempt this experiment for the BerkeleyParser since training was too slow.

Parser	Click as N	Click as V	Imperative rate
case-sensitive shift-reduce WSJ-only	74	0	6.3%
case-sensitive shift-reduce + Web-Train targeted self-training	75	0	10.5%
case-insensitive shift-reduce WSJ-only	75	0	10.3%
case-insensitive shift-reduce + Web-Train targeted self-training	75	0	11.6%
Berkeley WSJ-only	35	35	11.9%
Berkeley + Web-Train targeted self-training	13	58	12.5%
(WSJ-Train)	1	0	0.7%

Table 3: Counts on Web-Test of "click" tagged as a noun and verb and percentage of sentences parsed imperatively.

poses very different challenges compared to parsing newswire. We show how our method improves parsing performance and reordering performance on two examples: the trendy word "click" and imperative sentences.

5.3.1 Click

The word "click" appears only once in the training portion of the WSJ (as a noun), but appears many times in our Web test data. Table 3 shows the distribution of part-of-speech tags that different parsers assign to "click". The WSJ-only parsers tag "click" as a noun far too frequently. The WSJ-only shift-reduce parser refuses to tag "click" as a verb even with targeted self-training, but BerkeleyParser does learn to tag "click" more often as a verb.

It turns out that the shift-reduce parser's stubbornness is not due to a fundamental problem of the parser, but due to an artifact in TnT. To increase speed, TnT restricts the choices of tags for known words to previously-seen tags. This causes the parser's n-best lists to never hypothesize "click" as a verb, and self-training doesn't click no matter how targeted it is. This shows that the targeted self-training approach heavily relies on the diversity of the baseline parser's n-best lists.

It should be noted here that it would be easy to combine our approach with the *uptraining* approach of Petrov et al. (2010). The idea would be to use the BerkeleyParser to generate the n-best lists; perhaps we could call this *targeted uptraining*. This way, the shift-reduce parser could benefit both from the generally higher quality of the parse trees produced by the BerkeleyParser, as well as from the information provided by the extrinsic scoring function.

5.3.2 Imperatives

As Table 3 shows, the WSJ training set contains only 0.7% imperative sentences.[5] In contrast, our test sentences from the web contain approximately 10% imperatives. As a result, parsers trained exclusively on the WSJ underproduce imperative parses, especially a case-sensitive version of the baseline. Targeted self-training helps the parsers to predict imperative parses more often.

Targeted self-training works well for generating training data with correctly-annotated imperative constructions because the reordering of main subjects and verbs in an SOV language like Japanese is very distinct: main subjects stay at the beginning of the sentence, and main verbs are reordered to the end of the sentence. It is thus especially easy to know whether an imperative parse is correct or not by looking at the reference reordering. Figure 1 gives an example: the bad (WSJ-only) parse doesn't catch on to the imperativeness and gets a low reordering score.

6 Targeted Self-Training vs Training on Treebanks for Domain Adaptation

If task-specific annotation is cheap, then it is reasonable to consider whether we could use targeted self-training to adapt a parser to a new domain as a cheaper alternative to making new treebanks. For example, if we want to build a parser that can reorder question sentences better than our baseline WSJ-only parser, we have these two options:

1. Manually construct PTB-style trees for 2000

[5] As an approximation, we count every parse that begins with a root verb as an imperative.

questions and train on the resulting treebank.

2. Create reference reorderings for 2000 questions and then do targeted self-training.

To compare these approaches, we created reference reordering data for our train (2000 sentences) and test (1000 sentences) splits of the Question Treebank (Judge et al., 2006). Table 4 shows that both ways of training on QTB-Train sentences give similarly large improvements in reordering score on QTB-Test. Table 5 confirms that this corresponds to very large increases in English→Japanese BLEU score and subjective translation quality. In the human side-by-side comparison, the baseline translations achieved an average score of 2.12, while the targeted self-training translations received a score of 2.94, where a score of 2 corresponds to "some meaning/grammar" and "4" corresponds to "most meaning/grammar".

But which of the two approaches is better? In the shift-reduce parser, targeted self-training gives higher reordering scores than training on the treebank, and in BerkeleyParser, the opposite is true. Thus both approaches produce similarly good results. From a practical perspective, the advantage of targeted self-training depends on whether the extrinsic metric is cheaper to calculate than treebanking. For MT reordering, making reference reorderings is cheap, so targeted self-training is relatively advantageous.

As before, we can examine whether labeled attachment score measured on the test set of the QTB is predictive of reordering quality. Table 4 shows that targeted self-training raises LAS from 64.78→69.17%. But adding the treebank leads to much larger increases, resulting in an LAS of 84.75%, without giving higher reordering score. We can conclude that high LAS is not necessary to achieve top reordering scores.

Perhaps our reordering rules are somehow deficient when it comes to reordering correctly-parsed questions, and as a result the targeted self-training process steers the parser towards producing pathological trees with little intrinsic meaning. To explore this possibility, we computed reordering scores after reordering the QTB-Test treebank trees directly. Table 4 shows that this gives reordering scores similar to those of our best parsers. Therefore it is at least possible that the targeted self-training process could have resulted in a parser that achieves high reordering score by producing parses that look like those in the QuestionBank.

7 Related Work

Our approach to training parsers for reordering is closely related to self/up-training (McClosky. et al., 2006; Petrov et al., 2010). However, unlike uptraining, our method does not use only the 1-best output of the first-stage parser, but has access to the n-best list. This makes it similar to the work of McClosky. et al. (2006), except that we use an extrinsic metric (MT reordering score) to select a high quality parse tree, rather than a second, reranking model that has access to additional features.

Targeted self-training is also similar to the retraining of Burkett et al. (2010) in which they jointly parse unannotated bilingual text using a multiview learning objective, then retrain the monolingual parser models to include each side of the jointly parsed bitext as monolingual training data. Our approach is different in that it doesn't use a second parser and bitext to guide the creation of new training data, and instead relies on n-best lists and an extrinsic metric.

Our method can be considered an instance of weakly or distantly supervised structured prediction (Chang et al., 2007; Chang et al., 2010; Clarke et al., 2010; Ganchev et al., 2010). Those methods attempt to learn structure models from related external signals or aggregate data statistics. This work differs in two respects. First, we use the external signals not as explicit constraints, but to compute an oracle score used to re-rank a set of parses. As such, there are no requirements that it factor by the structure of the parse tree and can in fact be any arbitrary metric. Second, our final objective is different. In weakly/distantly supervised learning, the objective is to use external knowledge to build better structured predictors. In our case this would mean using the reordering metric as a means to train better dependency parsers. Our objective, on the other hand, is to use the extrinsic metric to train parsers that are specifically better at the reordering task, and, as a result, better suited for MT. This makes our work more in the spirit of Liang et al. (2006), who train a per-

Parser	QTB-Test reordering	QTB-Test LAS
Shift-reduce WSJ baseline	0.663	64.78%
+ treebank 1x	0.704	77.12%
+ treebank 10x	**0.768**	84.75%
+ targeted self-training 1x	0.746	67.84%
+ targeted self-training 10x	**0.779**	69.17%
Berkeley WSJ baseline	0.733	76.50%
+ treebank 1x	**0.800**	87.79%
+ targeted self-training 1x	**0.775**	80.64%
(using treebank trees directly)	0.788	100%

Table 4: Reordering and labeled attachment scores on QTB-Test for treebank training and targeted self-training on QTB-Train.

English to	QTB-Test BLEU		Human evaluation (scores range 0 to 6)		
	WSJ-only	Targeted	WSJ-only	Targeted	Sig. difference?
Japanese	0.2379	0.2615	2.12	2.94	yes (at 95% level)

Table 5: BLEU scores and human evaluation results for English→Japanese translation of the QTB-Test corpus, varying only the parser between systems between the WSJ-only shift-reduce parser and the QTB-Train targeted self-training 10x shift-reduce parser.

ceptron model for an end-to-end MT system where the alignment parameters are updated based on selecting an alignment from a n-best list that leads to highest BLEU score. As mentioned earlier, this also makes our work similar to Hall et al. (2011) who train a perceptron algorithm on multiple objective functions with the goal of producing parsers that are optimized for extrinsic metrics.

It has previously been observed that parsers often perform differently for downstream applications. Miyao et al. (2008) compared parser quality in the biomedical domain using a protein-protein interaction (PPI) identification accuracy metric. This allowed them to compare the utility of extant dependency parsers, phrase structure parsers, and deep structure parsers for the PPI identification task. One could apply the targeted self-training technique we describe to optimize any of these parsers for the PPI task, similar to how we have optimized our parser for the MT reordering task.

8 Conclusion

We introduced a variant of self-training that targets parser training towards an extrinsic evaluation metric. We use this targeted self-training approach to train parsers that improve the accuracy of the word reordering component of a machine translation system. This significantly improves the subjective quality of the system's translations from English into three SOV languages. While the new parsers give improvements in these external evaluations, their intrinsic attachment scores go down overall compared to baseline parsers trained only on treebanks. We conclude that when using a parser as a component of a larger external system, it can be advantageous to incorporate an extrinsic metric into parser training and evaluation, and that targeted self-training is an effective technique for incorporating an extrinsic metric into parser training.

References

A. Birch and M. Osborne. 2010. LRscore for evaluating lexical and reordering quality in MT. In *ACL-2010 WMT*.

T. Brants. 2000. TnT – a statistical part-of-speech tagger. In *ANLP '00*.

D. Burkett and D. Klein. 2008. Two languages are better than one (for syntactic parsing). In *EMNLP '08*.

D. Burkett, S. Petrov, J. Blitzer, and D. Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *CoNLL '10*.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL '07*.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. 2010. Structured output learning with indirect supervision. In *ICML '10*.

E. Charniak. 2000. A maximum–entropy–inspired parser. In *NAACL '00*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL '10*.

M. Collins, P. Koehn, and I. Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL '05*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL '97*.

M.-C. de Marneffe, B. MacCartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC '06*.

J. Foster. 2010. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *NAACL '10*.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *HLT-NAACL '04*.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

D. Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *COLING '10*.

D. Gildea. 2001. Corpus variation and parser performance. In *EMNLP '01*.

N. Habash. 2007. Syntactic preprocessing for statistical machine translation. In *MTS '07*.

K. Hall, R. McDonald, J. Katz-Brown, and M. Ringgaard. 2011. Training dependency parsers by jointly optimizing multiple objectives. In *EMNLP '11*.

H. Isozaki, T. Hirao, K. Duh, K. Sudoh, and H. Tsukada. 2010a. Automatic evaluation of translation quality for distant language pairs. In *EMNLP '10*.

H. Isozaki, K. Sudoh, H. Tsukada, and K. Duh. 2010b. Head finalization: A simple reordering rule for SOV languages. In *ACL-2010 WMT*.

J. Judge, A. Cahill, and J. v. Genabith. 2006. QuestionBank: creating a corpus of parse-annotated questions. In *ACL '06*.

A. Lavie and M. Denkowski. 2009. The Meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3).

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL '06*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

D. McClosky., E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *NAACL '06*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *ACL '05*.

Y. Miyao, R. Sætre, K. Sagae, T. Matsuzaki, and J. Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *ACL '08*.

J. Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4).

F. Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL '02*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *NAACL '07*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL '06*.

S. Petrov, P. Chang, and M. Ringgaard H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *EMNLP '10*.

D. Talbot, H. Kazawa, H. Ichikawa, J. Katz-Brown, M. Seno, and F. Och. 2011. A lightweight evaluation framework for machine translation reordering. In *EMNLP-2011 WMT*.

C. Wang. 2007. Chinese syntactic reordering for statistical machine translation. In *EMNLP '07*.

F. Xia and M. McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Coling '04*.

P. Xu, J. Kang, M. Ringgaard, and F. Och. 2009. Using a dependency parser to improve SMT for subject-object-verb languages. In *NAACL-HLT '09*.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *ACL '01*.

R. Zens and H. Ney. 2006. Discriminative reordering models for statistical machine translation. In *NAACL-06 WMT*.

Y. Zhang and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *EMNLP '08*.

Inducing Sentence Structure from Parallel Corpora for Reordering

John DeNero
Google Research
denero@google.com

Jakob Uszkoreit
Google Research
uszkoreit@google.com

Abstract

When translating among languages that differ substantially in word order, machine translation (MT) systems benefit from syntactic pre-ordering—an approach that uses features from a syntactic parse to permute source words into a target-language-like order. This paper presents a method for inducing parse trees automatically from a parallel corpus, instead of using a supervised parser trained on a treebank. These induced parses are used to pre-order source sentences. We demonstrate that our induced parser is effective: it not only improves a state-of-the-art phrase-based system with integrated reordering, but also approaches the performance of a recent pre-ordering method based on a supervised parser. These results show that the syntactic structure which is relevant to MT pre-ordering can be learned automatically from parallel text, thus establishing a new application for unsupervised grammar induction.

1 Introduction

Recent work in statistical machine translation (MT) has demonstrated the effectiveness of *syntactic pre-ordering*: an approach that permutes source sentences into a target-like order as a pre-processing step, using features of a source-side syntactic parse (Collins et al., 2005; Xu et al., 2009). Syntactic pre-ordering is particularly effective at applying structural transformations, such as the ordering change from a subject-verb-object (SVO) language like English to a subject-object-verb (SOV) language like Japanese. However, state-of-the-art pre-ordering methods require a supervised syntactic parser to provide structural information about each sentence. We propose a method that learns both a parsing model and a reordering model directly from a word-aligned parallel corpus. Our approach, which we call Structure Induction for Reordering (STIR), requires no syntactic annotations to train, but approaches the performance of a recent syntactic pre-ordering method in a large-scale English-Japanese MT system.

STIR predicts a pre-ordering via two pipelined models: (1) parsing and (2) tree reordering. The first model induces a binary parse, which defines the space of possible reorderings. In particular, only trees that properly separate verbs from their object noun phrases will license an SVO to SOV transformation. The second model locally permutes this tree. Our approach resembles work with binary synchronous grammars (Wu, 1997), but is distinct in its emphasis on monolingual parsing as a first phase, and in selecting reorderings without the aid of a target-side language model.

The parsing model is trained to maximize the conditional likelihood of trees that license the reorderings implied by observed word alignments in a parallel corpus. This objective differs from those of previous grammar induction models, which typically focus on succinctly explaining the observed source language corpus via latent hierarchical structure (Pereira and Schabes, 1992; Klein and Manning, 2002). Our convex objective allows us to train a feature-rich log-linear parsing model, even without supervised treebank data.

Focusing on pre-ordering for MT leads to a new

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 193–203,
Edinburgh, Scotland, UK, July 27–31, 2011.

perspective on the canonical NLP task of grammar induction—one which marries the wide-spread scientific interest in unsupervised parsing models with a clear application and extrinsic evaluation methodology. To support this perspective, we highlight several avenues of future research throughout the paper.

We evaluate STIR in a large-scale English-Japanese machine translation system. We measure how closely our predicted reorderings match those implied by hand-annotated word alignments. STIR approaches the performance of the state-of-the-art pre-ordering method described in Genzel (2010), which learns reordering rules for supervised treebank parses. STIR gives a translation improvement of 3.84 BLEU over a standard phrase-based system with an integrated reordering model.

2 Parsing and Reordering Models

STIR consists of two pipelined log-linear models for parsing and reordering, as well as a third model for inducing trees from parallel corpora, trees that serve to train the first two models. This section describes the domain and structure of each model, while Section 3 describes features and learning objectives.

Figure 1 depicts the relationship between the three models. For each aligned sentence pair in a parallel corpus, the *parallel parsing model* selects a binary tree t over the source sentence, such that t licenses the reordering pattern implied by the word alignment (Section 2.2). The *monolingual parsing model* is trained to generate t without inspecting the alignments or target sentences (Section 2.3). The *tree reordering model* is trained to locally permute t to produce the target order (Section 2.4). In the context of an MT system, the monolingual parser and tree reorderer are applied in sequence to pre-order source sentences.

2.1 Unlabeled Binary Trees

Unlabeled binary trees are central to the STIR pipeline. We represent trees via their constituent spans. Let $[k, \ell)$ denote a span of indices of a 0-indexed word sequence $\mathbf{e}$, where $i \in [k, \ell)$ if $k \leq i < \ell$. $[0, n)$ denotes the root span covering the whole sequence, where $n = |\mathbf{e}|$.

A tree $t = (\mathcal{T}, \mathcal{N})$ consists of a set of terminal spans $\mathcal{T}$ and non-terminal spans $\mathcal{N}$. Each non-terminal span $[k, \ell) \in \mathcal{N}$ has a split point m, where $k < m < \ell$ splits the span into child spans $[k, m)$ and $[m, \ell)$. Formally, a pair $(\mathcal{T}, \mathcal{N})$ is a well-formed tree over $[0, n)$ if:

- The root span $[0, n) \in \mathcal{T} \cup \mathcal{N}$.
- For each $[k, \ell) \in \mathcal{N}$, there exists exactly one m such that $\{[k, m), [m, \ell)\} \subset \mathcal{T} \cup \mathcal{N}$.
- Terminal spans $\mathcal{T}$ are disjoint, but cover $[0, n)$.

These trees include multi-word terminal spans. It is often convenient to refer to a split non-terminal triple (k, m, ℓ) that include a non-terminal span $[k, \ell)$ and its split point m. We denote the set of these triples as

$$\mathcal{N}^+ = \{(k, m, \ell) : \{[k, \ell), [k, m), [m, \ell)\} \in \mathcal{T} \cup \mathcal{N}\}.$$

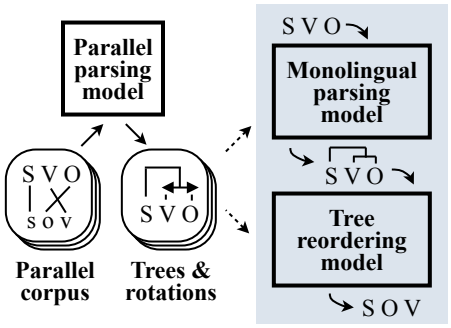


Figure 1: The training and reordering pipeline for STIR contains three models. The inputs and outputs of each model are indicated by solid arrows, while dashed arrows indicate the source of training examples. The parallel parsing model provides tree and reordering examples that are used to train the other models. In an MT system, the trained reordering pipeline (shaded) pre-orders a source sentence without target-side or alignment information.

2.2 Parallel Parsing Model

The first step in the STIR pipeline is to select a binary parse of each source sentence in a parallel corpus, one which licenses the reordering implied by a word alignment. Let the triple $(\mathbf{e}, \mathbf{f}, \mathcal{A})$ be an aligned sentence pair, where $\mathbf{e}$ and $\mathbf{f}$ are word sequences and $\mathcal{A}$ is a set of links (i, j) indicating that e_i aligns to f_j.

The set $\mathcal{A}$ provides ordering information over $\mathbf{e}$. To simplify definitions below, we first adjust $\mathcal{A}$ to

ignore all unaligned words in $\mathbf{f}$.

$$\mathcal{A}' = \{(i, c(j)) : (i,j) \in \mathcal{A}\}$$
$$c(j) = |\{j' : j' < j \ \wedge \ \exists i \text{ such that } (i,j') \in \mathcal{A}\}| \ .$$

$c(j)$ is the number of aligned words in $\mathbf{f}$ prior to position j. Next, we define a projection function:

$$\psi(i) = \left[\min_{j \in J_i} j, \max_{j \in J_i} j + 1\right)$$
$$J_i = \{j : (i,j) \in \mathcal{A}'\} \ ,$$

and let $\psi(i) = \emptyset$ if e_i is unaligned. We can extend this projection function to spans $[k, \ell)$ of $\mathbf{e}$ via union:

$$\psi(k, \ell) = \bigcup_{k \leq i < \ell} \psi(i) \ .$$

We say that a span $[k, \ell)$ aligns contiguously if

$$\forall (i,j) \in \mathcal{A}', j \in \psi(k,\ell) \text{ implies } i \in [k, \ell) \ ,$$

which corresponds to the familiar definition that $[k, \ell)$ is one side of an extractable phrase pair. Unaligned spans do not align contiguously.

Given this notion of projection, we can relate trees to alignments. A tree $(\mathcal{T}, \mathcal{N})$ over $\mathbf{e}$ respects an alignment $\mathcal{A}'$ if all $[k, \ell) \in \mathcal{T} \cup \mathcal{N}$ align contiguously, and for every (k, m, ℓ), the projections $\psi(k, m)$ and $\psi(m, \ell)$ are adjacent. Projections are adjacent if the left bound of one is the right bound of the other, or if either is empty.

The parallel parsing model is a linear model over trees that respect $\mathcal{A}'$, which factors over spans.

$$\mathrm{s}(t) = \sum_{[k,\ell) \in \mathcal{T}} w_T \phi_T(k, \ell) + \sum_{(k,m,\ell) \in \mathcal{N}^+} w_N \phi_N(k, m, \ell)$$

where the weight vector $w = (w_T \ w_N)$ scores features ϕ_T on terminal spans and ϕ_N on non-terminal spans and their split points.

Exact inference under this model can be performed via a dynamic program that exploits the following recurrence. Let $\mathrm{s}(k, \ell)$ be the score of the highest scoring binary tree over the span $[k, \ell)$ that respects $\mathcal{A}'$. Then,

$$\mathrm{s}_T(k, \ell) = \begin{cases} w_T \phi_T(k, \ell) & \text{if } [k, \ell) \text{ aligns contiguously} \\ -\infty & \text{otherwise} \end{cases}$$
$$f(k, m, \ell) = \mathrm{s}(k, m) + \mathrm{s}(m, \ell) + w_N \phi_N(k, m, \ell)$$
$$\mathrm{s}_N(k, \ell) = \max_{m : k < m < \ell} \begin{cases} f(k, m, \ell) & \text{if } \psi(k,m) \text{ is adjacent to } \psi(m, \ell) \\ -\infty & \text{otherwise} \end{cases}$$
$$\mathrm{s}(k, \ell) = \max\left[\mathrm{s}_T(k, \ell), \mathrm{s}_N(k, \ell)\right]$$

Above, s_T scores terminal spans while filtering out those which are not contiguous. The function f scores non-terminal spans by the sum of their child scores and additional features ϕ_N of the parent span. The recursive function s_N maximizes over split points while filtering out non-adjacent children. The recurrence will assign a score of $-\infty$ to any tree that does not respect $\mathcal{A}'$. Section 3 describes the features of this model. $\mathrm{s}(k, \ell)$ can be computed efficiently using the CKY algorithm.

2.3 Monolingual Parsing Model

The monolingual parsing model is trained to select the same trees as the parallel model, but without any features or constraints that reference word alignments. Hence, it can be applied to a source sentence before its translation is known.

This model also scores untyped binary trees according to a linear model parameterized by some $w = (w_T \ w_N)$ that weights features on terminal and non-terminal spans, respectively. We impose a maximum terminal length of L, but otherwise allow any binary tree. The score $\mathrm{s}(k, \ell)$ of the maximal tree over a span $[k, \ell)$ satisfies the familiar recurrence:

$$\mathrm{s}_M(k, \ell) = \begin{cases} w_T \phi_T(k, \ell) & \text{if } \ell - k \leq L \\ -\infty & \text{otherwise} \end{cases}$$
$$\mathrm{s}(k, \ell) = \max\left[\mathrm{s}_L(k, \ell), \max_{m : k < m < \ell} f(k, m, \ell)\right]$$

Inference under this recurrence can also be performed using the CKY algorithm. Section 3 describes the feature functions and training method.

2.4 Tree Reordering Model

Given a binary tree $(\mathcal{T}, \mathcal{N})$ over a sentence $\mathbf{e}$, we can reorder $\mathbf{e}$ by (a) permuting the children of non-terminals and (b) permuting the words of terminal spans. Formally, a reordering r assigns each terminal $[k, \ell) \in \mathcal{T}$ a permutation $\sigma(k, \ell)$ of its words and each split non-terminal (k, m, ℓ) a permutation $b(k, m, \ell)$ of its subspans, which can be either monotone or inverted, in the case of a binary tree. The permutation $\sigma(k, \ell)$ of a non-terminal span $[k, \ell) \notin \mathcal{T}$ is defined recursively as:

$$\begin{cases} \sigma(k, m)\ \sigma(m, \ell) & \text{if } b(k, m, \ell) \text{ is monotone} \\ \sigma(m, \ell)\ \sigma(k, m) & \text{if } b(k, m, \ell) \text{ is inverted} \end{cases}$$

In this paper, we use a reordering model that selects each terminal $\sigma(k, \ell)$ and each split non-terminal $b(k, m, \ell)$ independently, conditioned on the sentence $\mathbf{e}$. While the sub-problems of choosing $\sigma(k, \ell)$ and $b(k, m, \ell)$ are formally similar, we consider and evaluate them separately because the former deals only with local reordering, while the latter involves long-distance structural reordering.

Because our trees are binary, selecting $b(k, m, \ell)$ is a binary classification problem. Selecting $\sigma(k, \ell)$ for a terminal is a multiclass prediction problem that chooses among the $(\ell - k)!$ permutations of terminal $[k, \ell)$. Development experiments in English-Japanese yielded the best results with a maximum terminal span length $L = 2$. Hence, in experiments, terminal reordering is also binary classification.

Because each permutation is independent of all the others, reordering inference via a single pass through the tree is optimal. However, a more complex search procedure would be necessary to maintain optimality if the decision of $b(k, m, \ell)$ referenced other permutations, such as $\sigma([k, m))$ or $\sigma([m, \ell))$. Coupling together inference in this way represents a possible area of future study.

3 Features and Training Objectives

Each of these linear models factors over features on either terminal spans $[k, \ell)$ or split non-terminals (k, m, ℓ). Features vary in concert with the learning objectives and search spaces of each model.

Figure 2 shows an example sentence from our development corpus, including the target (Japanese) sentence, alignment, projections, parallel parser prediction, monolingual parser prediction, and predicted permutation. The feature descriptions below reference this example.

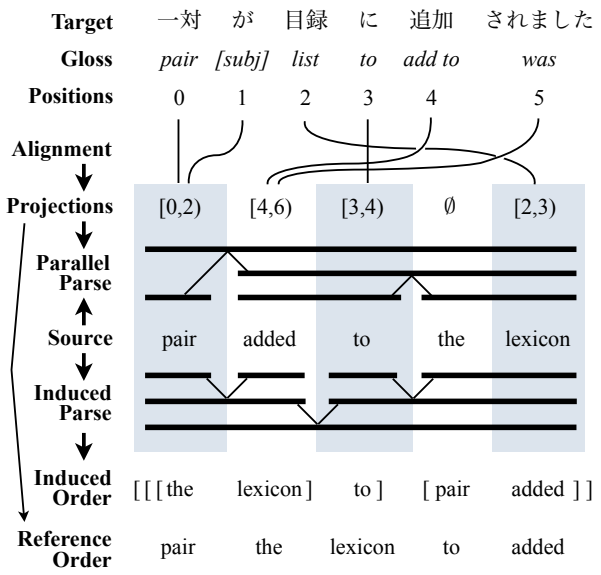


Figure 2: An example from our development corpus, annotated with the information flow (left) and annotations and predictions (right). Alignments inform projections, which are spans of the target associated with each source word. The parallel parse may only include contiguous spans. On the other hand, the induced parse may only condition on the source sentence. The induced order is restricted by the induced parse. In this example, the induced order is incorrect because the subject and verb form a constituent in the induced parse that cannot be separated correctly by the reordering model. This example demonstrates the important role of the induced parser in the STIR pipeline.

3.1 Tree Reordering Features

The tree reordering model consists of two local classifiers: the first can invert the two children of a non-terminal span, while the second can permute the words of a terminal span. The non-terminal classifier is trained on the trees that are selected by the parallel parsing model; the weights are chosen to minimize log loss of the correct permutation of each span (i.e., a maximum entropy model).

The terminal model is a multi-class maximum entropy model over the $n!$ possible permutations of the words in a terminal span. To make reordering more robust to monolingual parsing errors, the terminal

model is trained on all contiguous spans of each sentence up to length L, not just the terminal spans included in the parallel parsing tree.

The feature templates we apply to each span can be divided into the following five categories. Most features are shared across the two models.

Statistics. From a large aligned parallel corpus, we compute two statistics.

- $\mathrm{P}_C(e) = \frac{\text{count(e aligns contiguously)}}{\text{count}(e)}$ is the fraction of the time that a phrase e aligns contiguously to some target phrase, for all phrases up to length 4.
- $\mathrm{P}_D(e_i, e_j)$ is the fraction of the time that two co-occuring source words e_i and e_j align to adjacent positions in the target.

The first statistic indicates whether a contiguous phrase in the source should stay contiguous after reordering. Features based on this statistic apply to both terminal and short non-terminal spans. The second statistic indicates when a possibly discontiguous pair of words should be adjacent after reordering. This statistic is applied to pairs of words that would end up adjacent after an inversion: e_k and $e_{\ell-1}$ for span $[k, \ell)$. For instance, $\mathrm{P}_C(\text{added to}) = 0.68$ and $\mathrm{P}_D(\text{lexicon}, \text{to}) = 0.19$.

Cluster. All source word types are clustered into word classes, which together maximize likelihood of the source side of a large parallel corpus under a hidden Markov model, as in Uszkoreit and Brants (2008). Indicator features based on clusterings over c classes are defined over words e_k, e_{m-1}, e_m and $e_{\ell-1}$, as well as word sequences for spans up to length 4. Features are included for a variety of clusterings with sizes $c \in \{2^3, 2^4, \ldots, 2^{11}\}$.

POS. A supervised part-of-speech (POS) tagger provides coarse tags drawn from a 12 tag set $T = \{$Verb, Noun, Pronoun, Conjunction, Adjective, Adverb, Adposition, Determiner, Number, Particle/Function word, Punctuation, Other$\}$ (Petrov et al., 2011). Features based on these tags are computed identically to the features based on word classes.

Lexical. For a list of very common words in the source language, we include lexical indicator features for the boundary words e_k and $e_{\ell-1}$. For instance, the word "to" triggers a reordering, as do prepositions in general.

Length. Length computed as $\ell - k$, length as a fraction of sentence length, and quantized length features all contribute structural information.

All features except POS are computed directly from aligned parallel corpora. The Cluster and POS features play a similar role of expressing reordering patterns over collections of similar words. The ablation study in Section 5 compares these two feature sets directly.

3.2 Monolingual Parsing Features

The monolingual parsing model is also trained discriminatively, but involves structured prediction, as in a conditional random field (Lafferty et al., 2001). Conditional likelihood objectives have proven effective for supervised parsers (Finkel et al., 2008; Petrov and Klein, 2008). Recall that the score of a tree $t = (\mathcal{T}, \mathcal{N})$ factors over spans.

$$\mathrm{s}(t) = \sum_{[k,\ell) \in \mathcal{T}} w_T \phi_T(k, \ell) + \sum_{[k,\ell) \in \mathcal{N}} w_N \phi_N(k, m, \ell)$$

$$\mathrm{P}(t|\mathbf{e}) = \frac{\exp\left[\mathrm{s}(t)\right]}{\sum_{(t') \in \mathcal{B}(\mathbf{e})} \exp\left[\mathrm{s}(t')\right]}$$

where $\mathcal{B}(\mathbf{e})$ is the set of well-formed trees over $\mathbf{e}$.

The parallel parsing model (Section 2.2) produces a tree over the source sentence of each aligned sentence pair; these trees serve as our training examples. We can maximize their conditional likelihood according to this model via gradient methods. Each tree t over sentence $\mathbf{e}$ has a cumulative feature vector of dimension $|w| = |w_T| + |w_N|$, formed by stacking the terminal and non-terminal vectors:

$$\phi(t, \mathbf{e}) = \left(\sum_{[k,\ell) \in \mathcal{T}} \phi_T(k, \ell) \qquad \sum_{[k,\ell) \in \mathcal{N}} \phi_N(k, m, \ell) \right)$$

The contribution to the gradient objective from a tree t for a sentence $\mathbf{e}$ is the difference between observed

and expected feature vectors.

$$\mathcal{L}(w) = \sum_{(t,\mathbf{e})} \log \mathrm{P}(t|\mathbf{e})$$

$$\nabla\mathcal{L}(w) = \sum_{(t,\mathbf{e})} \left[\phi(t,\mathbf{e}) - \sum_{t' \in \mathcal{B}(\mathbf{e})} \mathrm{P}(t'|\mathbf{e}) \cdot \phi(t',\mathbf{e}) \right]$$

The second term in the gradient—the expected feature vector—can be computed efficiently because the feature vector $\phi(t')$ decomposes over the spans of t'. In particular, the inside-outside algorithm provides the quantities needed to compute the posterior probability of each terminal span $[k,\ell)$ and each split non-terminal (k,m,ℓ). Let, $\alpha(k,\ell)$ and $\beta(k,\ell)$ be the outside and inside scores of a span, respectively, computed using a log-sum semiring. Then, the log probablility that a terminal span $[k,\ell)$ appears in the tree for $\mathbf{e}$ under the posterior distribution $\mathrm{P}(t|\mathbf{e})$ is $\alpha(k,\ell) + w_T\phi_T(k,\ell)$. Note that this terminal posterior does not include the inside score of the span.

The log probability that a non-terminal span $[k,\ell)$ appears with split point m is

$$\alpha(k,\ell) + \beta(k,m) + \beta(m,\ell) + w_N\phi_N(k,m,\ell)$$

By the linearity of expectations, the expected feature vector for $\mathbf{e}$ can be computed by averaging the feature vectors of each terminal and split non-terminal span, weighted by their posterior probabilities.

In future work, one may consider training this model to maximize the likelihood of an entire forest of trees, in order to maintain uncertainty over which tree licensed a particular alignment.

We are currently using l-BFGS to optimize this objective over a relatively small training corpus, for 35 iterations. For this reason, we only include lexical features for very common words. Distributed or online training algorithms would perhaps allow for more training data (and therefore more lexicalized features) to be used in the future.

The features of this parsing model share the same types as the tree reordering models, but vary in their definition. The differences stem primarily from the different purpose of the model: here, features are not meant to decide how to reorder the sentence, but instead how to bracket the sentence hierarchically so that it can be reordered.

In particular, terminal spans have features on the sequence of POS tags and word clusters they contain, while a split non-terminal (k,m,ℓ) is scored based on the tags/clusters of the following words and word pairs: $e_k, e_{m-1}, e_m, e_{\ell-1}, (e_k,e_m), (e_k,e_{\ell-1})$, and (e_{m-1},e_m). The head word of a constituent often appears at one of its boundary positions, and so these features provide a proxy for explicitly tracking constituent heads in a parser.

Context features also appear, inspired by the constituent-context model of Klein and Manning (2001). For a span $[k,\ell)$, we add indicator features on the POS tags and word clusters of the words (e_{k-1},e_ℓ) which directly surround the constituent.

Features based on the statistic $\mathrm{P}_C(e)$ are also scored in the parsing model on all spans of length up to 4.

Length features score various structural aspects of each non-terminal (k,m,ℓ), such as $\frac{m-k}{\ell-k}$, $\frac{m-k}{k-m}$, etc.

One particularly interesting direction for future work is to train a single parsing model that licenses the reordering for several different languages. We might expect that a reasonable syntactic bracketing of English would simultaneously license the head-final transformations necessary to produce a Japanese or Korean ordering, and also the verb-subject-object ordering of formal Arabic.[1]

3.3 Parallel Parsing Features

The parallel parsing model does not run at translation time, but instead provides training examples to the other two models. Hence, defining an appropriate learning objective for this model is more challenging.

In the end, we are interested in selecting trees that we can learn to reproduce without an alignment (via the monolingual parsing model) and which can be reordered reliably (via the tree reordering model). Note that by construction, any tree selected by the parallel parsing model can be reordered perfectly. However, some of those trees will be easier to reproduce and reorder than others.

[1] An astute reviewer pointed out that no binary tree over an S-V-O sentence can license both S-O-V and V-S-O orderings. Hence, parse trees that are induced for multilingual reordering will need n-ary branches.

3.3.1 Reordering Loss Function

In order to measure the effectiveness of a reordering pipeline, we would like a metric over permutations. Fortunately, permutation loss for machine translation is already an established component of the METEOR metric, called a fragmentation penalty (Lavie and Agarwal, 2007). We define a slight variant of METEOR's fragmentation penalty that ranges from 0 to 1.

Given a sentence $\mathbf{e}$, a reference permutation σ^* of $(0, \cdots, |\mathbf{e}| - 1)$, and a hypothesized permutation $\hat{\sigma}$, let chunks$(\hat{\sigma}, \sigma^*)$ be the minimum number of "chunks" in $\hat{\sigma}$: the number of elements in a partition of $\hat{\sigma}$ such that each contiguous subsequence is also contiguous in σ^*.

We can define the reordering score between two permutations in terms of chunks.

$$R(\hat{\sigma}, \sigma^*) = \frac{|\sigma^*| - \text{chunks}(\hat{\sigma}, \sigma^*)}{|\sigma^*| - 1} \quad (1)$$

If $\hat{\sigma} = \sigma^*$, then chunks$(\hat{\sigma}, \sigma^*) = 1$. If no two adjacent elements of $\hat{\sigma}$ are adjacent in σ^*, then chunks$(\hat{\sigma}, \sigma^*) = |\sigma|$. Hence, the metric defined by Equation 1 ranges from 0 to 1.

The reference permutation σ^* of a source sentence $\mathbf{e}$ can be defined from an aligned sentence pair $(\mathbf{e}, \mathbf{f}, \mathcal{A})$ by sorting the words e_i of $\mathbf{e}$ by the left bound of their projection $\psi(i)$. Null-aligned words are placed to the left of the next aligned word to their right in the original order.

The reordering-specific loss functions defined in Equation 1 has been shown to correlate with human judgements of translation quality, especially for language pairs with substantial reordering like English-Japanese (Talbot et al., 2011). Other reordering-specific loss functions also correlate with human judgements (Birch et al., 2010). Future research could experiment with alternative reordering-based loss functions, such as Kendall's Tau, as suggested by Birch and Osborne (2011).

3.3.2 Parallel Parsing Objective

We can train our reordering pipeline by dividing an aligned parallel corpus into two halves, A and B, where the monolingual parsing and tree reordering models are trained on A, and their effectiveness is evaluated on held-out set B. Then, the effectiveness of the parallel parsing model is best measured on B, given fully trained parsing and reordering models.

$$\sum_{(\mathbf{e},\sigma^*) \in B} R\left(\sigma\left(\arg\max_{t \in \mathcal{B}(\mathbf{e})} [w \cdot \phi(t)]\right), \sigma^*\right) \quad (2)$$

Evaluating this objective involves training the other two models. Therefore, we can only hope to optimize this objective directly over a small dimensional space, for instance using a grid search. For this reason, we currently only include 4 features in the parallel parsing model for a tree t:

1. The sum of $\log \text{P}_C(e)$ for all terminals e in t with length greater than 1.
2. The count of length-1 terminal spans in t.
3. The count of terminals of length greater than k.
4. An indicator feature of whether parentheses and brackets are balanced in each span.

The model weights of features 3 and 4 above are fixed to large negative constants to prefer terminal spans of length up to k and spans with balanced punctuation. The weight of feature 1 is fixed to 1, and weight 2 was set via line search to 0.3. Ties among trees were broken randomly.

Of course, the problem of selecting training trees need not be directly tied to the end task of reordering, as in Equation 2. Instead, we might consider selecting trees according to a likelihood objective on the source side of a parallel corpus, similar to how monolingual grammar induction models often optimize corpus likelihood. In such a case, we could imagine training models with far more parameters, but we leave this research direction to future work.

4 Related Work

Our approach to inducing hierarchical structure for pre-ordering relates to several areas of previous work, including other pre-ordering methods, reordering models more generally, and models for the unsupervised induction of syntactic structure.

4.1 Pre-Ordering Models

Our reordering pipeline is intentionally similar to approaches that use a treebank-trained supervised

parser to reorder source sentences at training and translation time (Xia and McCord, 2004; Collins et al., 2005; Lee et al., 2010). Given a supervised parser, a rule-based pre-ordering procedure can either be specified by hand (Xu et al., 2009) or learned automatically (Genzel, 2010). We consider our approach to be a direct extension of these approaches, but one which induces structure from parallel corpora rather than relying on a treebank.

Tromble (2009) show that some pre-ordering benefits can be realized without a parsing step at all, by instead casting pre-ordering as a permutation modeling problem. While not splitting the task of pre-ordering into parsing and tree rordering, that work shows that pre-ordering models can be learned directly from parallel corpora.

4.2 Integrated Reordering Models

Distortion models have been primary components in machine translation models since the advent of statistical MT (Brown et al., 1993). In modern systems, reordering models are integrated into decoders as additional features in a discriminative log-linear model, which also includes a language model, translation features, etc. In these cases, reordering models interact with the strong signal of a target-side language model. Because ordering prediciton is conflated with target-side generation, evaluations are conducted on the entire generated output, which cannot isolate reordering errors from other sorts of errors, like lexical selection.

Despite these differences, certain integrated reordering models are similar in character to syntactic pre-ordering models. In particular, the tree rotation model of Yamada and Knight (2001) posited that reordering decisions involve rotations of a source-side syntax tree. The parameters of such a model can be trained by treating tree rotations as latent variables in a factored translation model, which parameterizes reordering and transfer separately but performs joint inference (Dyer and Resnik, 2010). Syntactic reordering and transfer can also be modeled jointly, for instance in a tree-to-string translation system parameterized by a transducer grammar.

While the success of integrated reordering models certainly highlights the importance of reordering in machine translation systems, we see several advantages to a pipelined, pre-ordering approach. First, the pre-ordering model can be trained and evaluated directly. Second, pre-ordering models need not factor according to the same dynamic program as the translation model. Third, the same reordering can be applied during training (for word alignment and rule extraction) and translation time without adding complexity to the extraction and decoding algorithms. Of course, integrating our model into translation inference represents a potentially fruitful avenue of future research.

4.3 Grammar Induction

The language processing community actively works on the problem of automatically inducing grammatical structure from a corpus of text (Pereira and Schabes, 1992). Some success in this area has been demonstrated via generative models (Klein and Manning, 2002), which often benefit from well-chosen priors (Cohen and Smith, 2009) or posterior constraints (Ganchev et al., 2009). In principle, these models must discover the syntactic patterns that govern a language from the sequences of word tokens alone. These models are often evaluated relative to reference treebank annotations.

Grammar induction in the context of machine translation reordering offers different properties. The alignment patterns in a parallel corpus provide an additional signal to models that is strongly tied to syntactic properties of the aligned languages. Also, the evaluation is straightforward—any syntactic structure that supports the prediction of reordering is rewarded.

Kuhn (2004) applied alignment-based constraints to the problem of inducing probabilistic context-free grammars, and showed an improvement with respect to Penn Treebank annotations over monolingual induction. Their work is distinct from ours because it focused on projecting distituents across languages, but mirrors ours in demonstrating that there is a role for aligned parallel corpora in grammar induction.

Snyder et al. (2009) also demonstrated that parallel corpora can play a role in improving the quality of grammar induction models. Their work differs from ours in that it focuses on multilingual lexical statistics and dependency relationships, rather than reordering patterns.

		Parsing			Tree Reordering			Pipeline
		Prec	Rec	$F1$	acc_N	acc_T	R_O	R
Annotated word alignments	All features	82.0	87.8	84.8	97.3	93.6	87.7	80.5
	All but POS	81.3	87.7	84.4	97.0	92.6	86.6	79.4
	All but Cluster	81.2	87.9	84.4	95.9	93.2	83.8	77.8
	All but POS & Cluster	75.4	82.0	78.5	89.2	89.7	66.8	49.7
Learned alignments	All features	72.5	61.0	66.3	91.6	83.3	72.0	49.5
Monotone order								34.9
Inverted order								30.8
Syntactic pre-ordering (Genzel, 2010)								66.0

Table 1: Accuracy of individual monolingual parsing and reordering models, as well as complete pipelines trained on annotated and learned word alignments.

4.4 Bilingual Grammar Induction

Also related to STIR is previous work on bilingual grammar induction from parallel corpora using ITG (Blunsom et al., 2009). These models have focused on learning phrasal translations — which are the terminal productions of a synchronous ITG — rather than reordering patterns that occur higher in the tree. Hence, while this paper shares formal machinery and data sources with that line of work, the models themselves target orthogonal aspects of the translation problem.

5 Experimental Results

As training data for our models we used 14,000 English sentences that were sampled from the web, translated into Japanese, and manually annotated with word alignments. The annotation was carried out by the original translators to promote consistency of analysis. Talbot et al. (2011) describes this corpus in further detail. A held-out test set of 396 manually aligned sentence pairs was used to evaluate reordering accuracy. Statistics used for features were computed from the full, unreordered, automatically word aligned, parallel training corpus used for the translation experiments described below.

5.1 Individual Model Accuracy

We evaluate the accuracy of the monolingual parsing models by their span F1, relative to the trees induced by the parallel parsing model on the held-out set. The first row of Table 1 shows that the model was able to reliably replicate the parses induced from alignments, at 84.8% F1. The following three lines show that removing either POS or cluster features degrades performance by only 0.4% F1, indicating that POS features are largely redundant in the presence of automatically induced word class features. Hence, no syntactic annotations are necessary at all to train the model.

We report two accuracy measures for the tree reordering model, one for non-terminal spans (acc_N) and one for terminal spans (acc_T). The following column, labeled R_O, is the reordering score of the tree reordering model applied to the oracle parallel parser tree. This score is independent of the monolingual parsing model.

The fifth line, labeled *learned alignments*, shows the impact of replacing manual alignment annotations with learned Model 1 alignments, trained in both directions and combined with the *refined* heuristic (Brown et al., 1993; Och et al., 1999).

The *pipeline* column shows the reordering score of the full STIR pipeline compared to two simple baselines: *Monotone* applies no reordering, while *inverted* simply inverts the word order. STIR outperforms all three other systems.

In the final line, we compare to the syntax-based pre-ordering system described in Genzel (2010). This approach first parses source sentences with a supervised parser, then learns reordering rules that permute those trees.

5.2 Translation Quality

We apply STIR as a pre-ordering step in a state-of-the-art phrase-based translation system from English to Japanese (Koehn et al., 2003). At training

time, pre-ordering is applied to the source side of every sentence pair in the training corpus before word alignment and phrase extraction. Likewise, every input sentence is pre-ordered at translation time.

Our baseline is the same system, but without pre-ordering. Our implementation's integrated distortion model is expressed as a negative exponential function of the distance between the current and previous source phrase, with a maximum jump width of four words. Our in-house decoder is based on the alignment template approach to translation and uses a small set of standard feature functions during decoding (Och and Ney, 2004).

We compare to using an integrated lexicalized reordering model (Koehn and Monz, 2005), a forest-to-string translation model (Zhang et al., 2011) and finally the syntactic pre-ordering technique of Genzel (2010) applied to the phrase-based baseline. We evaluate the impact of the proposed approach on translation quality as measured by the BLEU score on the token level (Papineni et al., 2002).

The translation model is trained on 700 million tokens of parallel text, primarily extracted from the web using automated parallel document identification (Uszkoreit et al., 2010). Alignments were learned using two iterations of Model 1 and two iterations of the HMM alignment model (Vogel et al., 1996). Our *dev* and *test* data sets consist of 3100 and 1000 English sentences, respectively, that were randomly sampled from the web and translated into Japanese. The *eval* set is a larger, heterogenous set containing 12,784 sentences. In all cases, the final log-linear models were optimized on the *dev* set using lattice-based Minimum Error Rate Training (Macherey et al., 2008).

Table 2 shows that STIR improves over the baseline system by a large margin of 3.84% BLEU (test). These gains are comparable in magnitude to those reported in Genzel (2010). Our induced parses are competitive with both systems that use syntactic parsers and substantially outperform lexicalized reordering.

	BLEU %		
	dev	test	eval
Baseline	18.65	19.02	13.60
Lexicalized Reordering	19.45	18.92	13.99
Forest-to-String	**23.08**	22.85	**16.60**
Syntactic Pre-ordering	22.59	**23.28**	16.31
STIR: annotated	22.46	22.86	16.39
STIR: learned	20.28	20.66	14.64

Table 2: Translation quality, measured by BLEU, for English to Japanese. STIR results use both manually annotated and learned alignments.

6 Conclusion

We have demonstrated that induced parses suffice for pre-ordering. We hope that future work in grammar induction will also consider pre-ordering as an extrinsic evaluation.

References

Alexandra Birch and Miles Osborne. 2011. Reordering metrics for MT. In *Proceedings of the Association for Computational Linguistics*.

Alexandra Birch, Phil Blunsom, and Miles Osborne. 2010. Metrics for MT evaluation: Evaluating reordering. *Machine Translation*.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Association for Computational Linguistics*.

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

Shay Cohen and Noah Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the Association for Computational Linguistics*.

Chris Dyer and Philip Resnik. 2010. Context-free reordering, finite-state translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of the Association for Computational Linguistics*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Association for Computational Linguistics*.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the Conference on Computational Linguistics*.

Dan Klein and Christopher D. Manning. 2001. Natural language grammar induction using a constituent-context model. In *Proceedings of Neural Information Processing Systems*.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the Association for Computational Linguistics*.

Philipp Koehn and Christof Monz. 2005. Shared task: Statistical machine translation between european languages. In *Proceedings of the International Workshop on Spoken Language Translation*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Proceedings of the Association for Computational Linguistics*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of ACL Workshop on Statistical Machine Translation*.

Young-Suk Lee, Bing Zhao, and Xiaoqiang Luo. 2010. Constituent reordering and syntax models for English-to-Japanese statistical machine translation. In *Proceedings of the Conference on Computational Linguistics*.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*.

Franz Josef Och, Christopher Tillman, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the Association for Computational Linguistics*.

Slav Petrov and Dan Klein. 2008. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. Technical report.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Association for Computational Linguistics*.

David Talbot, Hideto Kazawa, Hiroshi Ichikawa, Jason Katz-Brown, Masakazu Seno, and Franz J. Och. 2011. A lightweight evaluation framework for machine translation reordering. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Roy Tromble. 2009. Learning linear ordering problems for better translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of the Association for Computational Linguistics*.

Jakob Uszkoreit, Jay Ponte, Ashok Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proceedings of the Conference on Computational Linguistics*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the Conference on Computational linguistics*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of the Conference on Computational Linguistics*.

Peng Xu, Jaeho Kang, Michael Ringgard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the Association for Computational Linguistics*.

Hao Zhang, Licheng Fang, Peng Xu, and Xiaoyun Wu. 2011. Binarized forest to string translation. In *Proceedings of the Association for Computational Linguistics*.

Augmenting String-to-Tree Translation Models with Fuzzy Use of Source-side Syntax

Jiajun Zhang, Feifei Zhai and Chengqing Zong
Institute of Automation, Chinese Academy of Sciences
Beijing, China
{jjzhang, ffzhai, cqzong}@nlpr.ia.ac.cn

Abstract

Due to its explicit modeling of the grammaticality of the output via target-side syntax, the string-to-tree model has been shown to be one of the most successful syntax-based translation models. However, a major limitation of this model is that it does not utilize any useful syntactic information on the source side. In this paper, we analyze the difficulties of incorporating source syntax in a string-to-tree model. We then propose a new way to use the source syntax in a fuzzy manner, both in source syntactic annotation and in rule matching. We further explore three algorithms in rule matching: 0-1 matching, likelihood matching, and deep similarity matching. Our method not only guarantees grammatical output with an explicit target tree, but also enables the system to choose the proper translation rules via fuzzy use of the source syntax. Our extensive experiments have shown significant improvements over the state-of-the-art string-to-tree system.

1 Introduction

In recent years, statistical translation models based upon linguistic syntax have shown promising progress in improving translation quality. It appears that encoding syntactic annotations on either side or both sides in translation rules can increase the expressiveness of rules and can produce more accurate translations with improved reordering.

One of the most successful syntax-based models is the string-to-tree model (Galley et al., 2006; Marcu et al., 2006; Shen et al., 2008; Chiang et al., 2009). Since it explicitly models the grammaticality of the output via target-side syntax, the string-to-tree model (Xiao et al., 2010) significantly outperforms both the state-of-the-art phrase-based system Moses (Koehn et al., 2007) and the formal syntax-based system Hiero (Chiang, 2007). However, there is a major limitation in the string-to-tree model: it does not utilize any useful source-side syntactic information, and thus to some extent lacks the ability to distinguish good translation rules from bad ones.

The source syntax is well-known to be helpful in improving translation accuracy, as shown especially by tree-to-string systems (Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006; Mi et al., 2008; Zhang et al., 2009). The tree-to-string systems are simple and efficient, but they also have a major limitation: they cannot guarantee the grammaticality of the translation output because they lack target-side syntactic constraints.

Thus a promising solution is to combine the advantages of the tree-to-string and string-to-tree approaches. A natural idea is the tree-to-tree model (Ding and Palmer, 2005; Cowan et al., 2006; Liu et al., 2009). However, as discussed by Chiang (2010), while tree-to-tree translation is indeed promising in theory, in practice it usually ends up over-constrained. Alternatively, Mi and Liu (2010) proposed to enhance the tree-to-string model with target dependency structures (as a language model). In this paper, we explore in the other direction: based on the strong string-to-tree model which builds an explicit target syntactic tree during decoding rather than apply only a syntactic language model, we aim to find a useful way to incorporate the source-side syntax.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 204–215,
Edinburgh, Scotland, UK, July 27–31, 2011.

First, we give a motivating example to show the importance of the source syntax for a string-to-tree model. Then we discuss the difficulties of integrating the source syntax into the string-to-tree model. Finally, we propose our solutions.

Figure 1 depicts a standard process that transforms a Chinese string into an English tree using several string-to-tree translation rules. The tree with solid lines is produced by the baseline string-to-tree system. Although the yield is grammatical, the translation is not correct since the system mistakenly applies rule r_2, thus translating the Chinese preposition 和(hé) in the example sentence into the English conjunction *and*. As a result, the Chinese prepositional phrase '和 恐怖 组织 网' ("with terrorist networks") is wrongly translated as a part of the relevant noun phrase ("[Hussein] and terrorists networks"). Why does this happen? We find that r_2 occurs 103316 times in our training data, while r_3 occurs only 1021 times. Thus, without source syntactic clues, the Chinese word 和(hé) is converted into the conjunction *and* in most cases. In general, this conversion is correct when the word 和(hé) is used as a conjunction. But 和(hé) is a preposition in the source sentence. If we are given this source syntactic clue, rule r_3 will be preferred. This example motivates us to provide a moderate amount of source-side syntactic information so as to obtain the correct English tree with dotted lines (as our proposed system does).

A natural question may arise that is it easy to incorporate source syntax in the string-to-tree model? To the best of our knowledge, no one has studied this approach before. In fact, it is not a trivial question if we look into the string-to-tree model. We find that the difficulties lie in at least three problems: 1) For a string-to-tree rule such as r_6 in figure 1, how should we syntactically annotate its source string? 2) Given the source-annotated string-to-tree rules, how should we match these rules according to the test source tree during decoding? 3) How should we binarize the source-annotated string-to-tree rules for efficient decoding?

For the first problem, one may require the source side of a string-to-tree rule to be a constituent. However, such excessive constraints will exclude many good string-to-tree rules whose source strings are not constituents. Inspired by Chiang (2010), we adopt a fuzzy way to label every source string with the complex syntactic categories of SAMT (Zollmann and Venugopal, 2006). This method leads to a one-to-one correspondence between the new rules and the string-to-tree rules. We will detail our fuzzy labeling method in Section 2.

For the second problem, it appears simple and intuitive to match rules by requiring a rule's source syntactic category to be the same as the category of the test string. However, this hard constraint will greatly narrow the search space during decoding. Continuing to pursue the fuzzy methodology, we adopt a fuzzy matching procedure to enable matching of all the rules whose source strings match the test string, and then determine the degree of matching between the test source tree and each rule. We will discuss three fuzzy matching algorithms, from simple to complex, in Section 3.

The third question is a technical problem, and we will give our solution in Section 4.

Our method not only guarantees the grammaticality of the output via the target tree structure, but also enables the system to choose appropriate translation rules during decoding through source syntactic fuzzy labeling and fuzzy matching.

The main contributions of this paper are as follows:

1) We propose a fuzzy method for both source syntax annotation and rule matching for augmenting string-to-tree models.
2) We design and investigate three fuzzy rule matching algorithms: 0-1 matching, likelihood matching, and deep similarity matching.

We hope that this paper will demonstrate how to effectively incorporate both source and target syntax into a translation model with promising results.

2 Rule Extraction

Since we annotate the source side of each string-to-tree rule with source parse tree information in a fuzzy way, we will henceforward denote the source-syntax-decorated string-to-tree rule as a **fuzzy-tree to exact-tree rule**. We first briefly review issues of string-to-tree rule extraction; then we discuss how to augment the string-to-tree rules to yield fuzzy-tree to exact-tree rules.

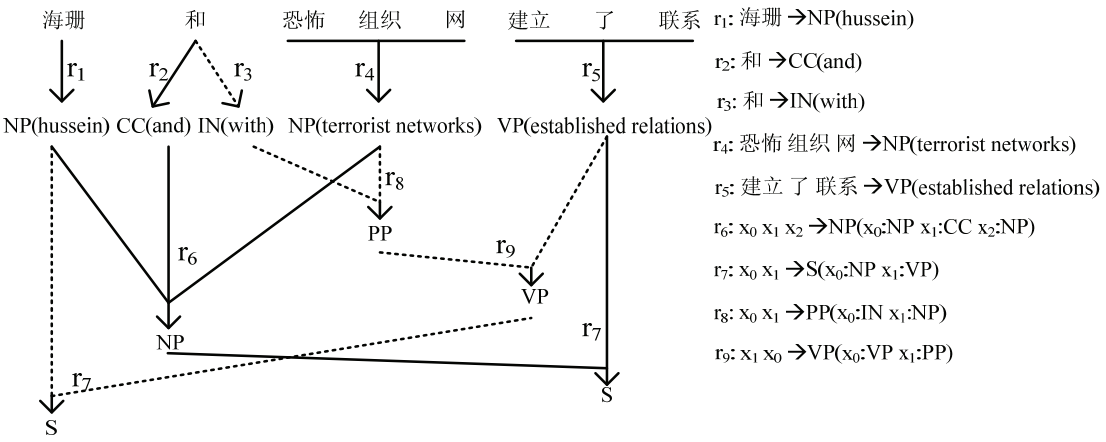


Figure 1: Two alternative derivations for a sample string-to-tree translation. The rules used are listed on the right. The target yield of the tree with solid lines is *hussein and terrorist networks established relations*. The target yield of the tree with dotted lines is *hussein established relations with terrorist networks*.

2.1 String-to-Tree Rule Extraction

Galley et al. (2004) proposed the GHKM algorithm for extracting (minimal) string-to-tree translation rules from a triple (*f, e_t, a*), where *f* is the source-language sentence, e_t is a target-language parse tree whose yield *e* is the translation of *f*, and *a* is the set of word alignments between *e* and *f*. The basic idea of GHKM is to obtain the set of minimally-sized translation rules which can explain the mappings between source string and target parse tree. The minimal string-to-tree rules are extracted in three steps: (1) frontier set computation; (2) fragmentation; and (3) extraction.

The frontier set (**FS**) is the set of potential points at which to cut the graph *G* constructed by the triple (*f, e_t, a*) into fragments. A node satisfying the word alignment is a frontier. ***Bold italic*** nodes in the English parse tree in Figure 2 are all frontiers.

Given the frontier set, a well-formed fragmentation of *G* is generated by restricting each fragment to take only nodes in **FS** as the root and leaf nodes.

With fragmentation completed, the rules are extracted through a depth-first traversal of e_t: for each frontier being visited, a rule is extracted. These extracted rules are called *minimal rules* (Galley et al., 2004). For example, rules $r_a \sim r_i$ in Figure 2 are part of the total of 13 minimal rules.

To improve the rule coverage, SPMT models can be employed to obtain *phrasal rules* (Marcu et at., 2006). In addition, the minimal rules which share the adjacent tree fragments can be connected together to form composed rules (Galley et al., 2006). In Figure 2, r_j is a rule composed by combining r_c and r_g.

2.2 Fuzzy-tree to Exact-tree Rule Extraction

Our fuzzy-tree to exact-tree rule extraction works on word-aligned tree-to-tree data (Figure 2 illustrates a Chinese-English tree pair). Basically, the extraction algorithm includes two parts:

(1) String-to-tree rule extraction (without considering the source parse tree);
(2) Decoration of the source side of the string-to-tree rules with syntactic annotations.

We use the same algorithm introduced in the previous section for extracting the base string-to-tree rules. The source-side syntactic decoration is much more complicated.

The simplest way to decorate, as mentioned in the Introduction, is to annotate the source-side of a string-to-tree rule with the syntactic tag that exactly covers the source string. This is what the exact tree-to-tree procedure does (Liu et al., 2009). However, many useful string-to-tree rules will become invalid if we impose such a tight restriction. For example, in Figure 2, the English phrase *discuss ... them* is a VP, but its Chinese counterpart is not a constituent. Thus we will miss the rule r_h although it is a useful reordering rule. According to the analysis of our training data, the rules with rigid source-side syntactic constraints account for only about 74.5% of the base string-to-tree rules. In this paper, we desire more general applicability.

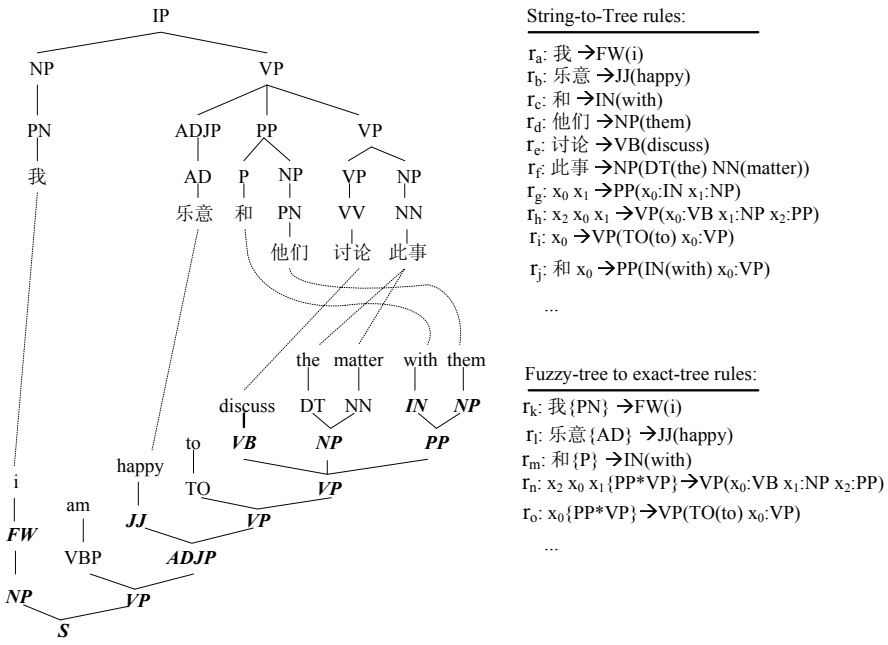


Figure 2: A sample Chinese-English tree pair for rule extraction. The ***bold italic*** nodes in the target English tree are frontiers. **Note that** string-to-tree rules are extracted without considering source-side syntax (upper-right). The new fuzzy-tree to exact-tree rules are extracted with both-side parse trees (bottom-right).

Inspired by (Zollmann and Venugopal, 2006; Chiang, 2010), we resort to SAMT-style syntactic categories in the style of categorial grammar (Bar-Hillel, 1953). The annotation of the source side of string-to-tree rules is processed in three steps: (1) If the source-side string corresponds to a syntactic category C in the source parse tree, we label the source string with C. (2) Otherwise, we check if there exists an extended category of the forms C_1*C_2, C_1/C_2 or $C_2\backslash C_1$[1], indicating respectively that the source string spans two adjacent syntactic categories, a partial syntactic category C_1 missing a C_2 on the right, or a partial C_1 missing a C_2 on the left. (3) If the second step fails, we check if there is an extended category of the forms C_1*C_2*C_3 or $C_1..C_2$, showing that the source string spans three adjacent syntactic categories or a partial category with C_1 and C_2 on each side. In the worst case, $C_1..C_2$ can denote every source string, thus all of the decorations in our training data can be explained within the above three steps. Using the SAMT-style grammar, each source string can be associated with a syntactic category. Thus our fuzzy-tree to exact-tree extraction does not lose any rules as compared with string-to-tree extraction. For example, rule r_o in Figure 2 uses the product category $PP*VP$ on the source side.

A problem may arise: How should we handle the situation where several rules are observed which only differ in their source-side syntactic categories? For example, besides the rule r_m in Figure 2, we encountered rules like $和\{CC\} \rightarrow IN(with)$ in the training data. Which source tag should we retain? We do not make a partial choice in the rule extraction phase. Instead, we simply make a union of the relevant rules and retain the respective tag counts. Applying this strategy, the rule takes the form of $和\{P:6, CC:4\} \rightarrow IN(with)$[2], indicating that the source-side preposition tag appears six times while the conjunction occurs four times. Note that the final rule format used in translation depends on the specific fuzzy rule matching algorithm adopted.

3 Fuzzy Rule Matching Algorithms

The extracted rules will ultimately be applied to derive translations during decoding. One way to apply the fuzzy-tree to exact-tree rules is to narrow the rule search space. Given a test source sentence

[1] The kinds of categories are checked in order. This means that if C_1*C_2, C_1/C_2 can both describe the same source string, we will choose C_1*C_2.

[2] 6 and 4 are not real counts. They are used for illustration only.

with its parse tree, we can according to this strategy choose only the rules whose source syntax matches the test source tree. However, this restriction will rule out many potentially correct rules. In this study, we keep the rule search space identical to that of the string-to-tree setting, and postpone the use of source-side syntax until the derivation stage. During derivation, a fuzzy matching algorithm will be adopted to compute a score to measure the compatibility between the rule and the test source syntax. The translation model will learn to distinguish good rules from bad ones via the compatibility scores.

In this section, three fuzzy matching algorithms, from simple to complex, are investigated in order.

3.1 0-1 Matching

0-1 matching is a straightforward approach that rewards rules whose source syntactic category exactly matches the syntactic category of the test string and punishes mismatches. It has mainly been employed in hierarchical phrase-based models for integrating source or both-side syntax (Marton and Resnik, 2008; Chiang et al., 2009; Chiang, 2010). Since it is verified to be very effective in hierarchical models, we borrow this idea in our source-syntax-augmented string-to-tree translation.

In 0-1 matching, the rule's source side must contain only one syntactic category, but a rule may have been decorated with more than one syntactic category on the source side. Thus we have to choose the most reliable category and discard the others. Here, we select the one with the highest frequency. For example, the tag *P* in the rule $和\{P:6, CC:4\} \rightarrow IN(with)$ appears more frequently, so the final rule used in 0-1 matching will be $和\{P\} \rightarrow IN(with)$. Accordingly, we design two features:

1. *match_count* calculates in a derivation the number of rules whose source-side syntactic category matches the syntactic category of the test string.
2. *unmatch_count* counts the number of mismatches.

For example, in the derivations of Figure 1, we know the Chinese word 和(hé) is a preposition in this sentence (and thus can be written as P(和)), therefore, *match_count += 1* if the above rule $和\{P\} \rightarrow IN(with)$ is employed.

These two features are integrated into the log-linear translation model and the corresponding feature weights will be tuned along with other model features to learn which rules are preferred.

3.2 Likelihood Matching

It appears intuitively that the 0-1 matching algorithm does not make full use of the source-side syntax because it keeps only the most-frequent syntactic label and discards some potentially useful information. Therefore, it runs the risk of treating all the discarded source syntactic categories of the rule as equally likely. For example, there is an extracted rule as follows:

$的\{DEC:11233, DEG:11073, DEV:65\} \rightarrow IN(of)$

0-1 matching converts it into $的\{DEC\} \rightarrow IN(of)$. The use of this rule will be penalized if the syntactic category of the test string 的(dē) is parsed as *DEG* or *DEV*. On one hand, the frequency of the tag *DEG* is just slightly less than that of *DEC*, but the 0-1 matching punishes the former while rewarding the latter. On the other hand, the frequency of *DEG* is much more than that of *DEV*, but they are penalized equally. It is obvious that the syntactic categories are not finely distinguished.

Considering this situation, we propose the likelihood matching algorithm. First, we compute the likelihood of the rule's source syntactic categories. Since we need to deal with the potential problem that the rule is hit by the test string but the syntactic category of the test string is not in the category set of the rule's source side, we apply the *m*-estimate of probability (Mitchell, 1997) to calculate a smoothed likelihood

$$likelihood_c = \frac{n_c + mp}{n + m} \quad (1)$$

in which n_c is the count of each syntactic category *c* in a specific rule, *n* denotes the total count of the rule, *m* is a constant called the equivalent sample size, and *p* is the prior probability of the category *c*. In our work, we set the constant *m=1* and the prior *p* to 1/12599 where 12599 is the total number of source-side syntactic categories in our training data. For example, the rule $和\{P:6, CC:4\} \rightarrow IN(with)$ becomes $和\{P:0.545, CC:0.364, 7.2e\text{-}6\} \rightarrow IN(with)$ after likelihood computation. Then, if we apply likelihood matching in the derivations in Figure 1 where the test string is 和 and its syntax is P(和),

the matching score with the above rule will be 0.545. When the test Chinese word 和 is parsed as a category other than *P* or *CC*, the matching score with the above rule will be 7.2e-6.

Similar to 0-1 matching, likelihood matching will serve as an additional model feature representing the compatibility between categories and rules.

3.3 Deep Similarity Matching

Considering the two algorithms above, we can see that the purpose of fuzzy matching is in fact to calculate a similarity. 0-1 matching assigns similarity 1 for exact matches and 0 for mismatch, while likelihood matching directly utilizes the likelihood to measure the similarity. Going one step further, we adopt a measure of deep similarity, computed using latent distributions of syntactic categories. Huang et al. (2010) proposed this method to compute the similarity between two syntactic tag sequences, used to impose soft syntactic constraints in hierarchical phrase-based models. Analogously, we borrow this idea to calculate the similarity between two SAMT-style syntactic categories, and then apply it to calculate the degree of matching between a translation rule and the syntactic category of a test source string for purposes of fuzzy matching. We call this procedure **deep similarity matching**.

Instead of directly using the SAMT-style syntactic categories, we represent each category by a real-valued feature vector. Suppose there is a set of *n* latent syntactic categories $V=(v_1,\cdots,v_n)$ (*n=16* in our experiments). For each SAMT-style syntactic category, we compute its distribution of latent syntactic categories $\vec{P}_c(V)=(P_c(v_1),\cdots,P_c(v_n))$. For example, $\vec{P}_{VP*NP}(V)=(0.4, 0.2, 0.3, 0.1)$ means that the latent syntactic categories v_1, v_2, v_3, v_4 are distributed as $p(v_1)$=0.4, $p(v_2)$=0.2, $p(v_3)$=0.3 and $p(v_4)$=0.1 for the SAMT-style syntactic category *VP*NP*. Then we further transform the distribution to a normalized feature vector $\vec{F}(c)=\vec{P}_c(V)/\left\|\vec{P}_c(V)\right\|$ to represent the SAMT-style syntactic category *c*.

With the real-valued vector representation for each SAMT-style syntactic category, the degree of similarity between two syntactic categories can be simply computed as a dot-product of their feature vectors:

$$\vec{F}(c)\cdot\vec{F}(c')=\sum_{1\le i\le n} f_i(c)f_i(c') \quad (2)$$

This computation yields a similarity score ranging from 0 (totally different syntactically) to 1 (totally identical syntactically).

Since we can now compute the similarity of any syntactic category pair, we are currently ready to compute the matching degree between the syntactic category of a test source string and a fuzzy-tree to exact-tree rule. To do this, we first convert the original fuzzy-tree to exact-tree rule to the rule of likelihood format without any smoothing. For example, the rule $和\{P:6, CC:4\}\rightarrow IN(with)$ becomes $和\{P:0.6, CC:0.4\}\rightarrow IN(with)$ after conversion. We then denote the syntax of a rule's source-side ***RS*** by weighting all the SAMT-style categories in RS

$$\vec{F}(RS)=\sum_{c\in RS} P_{RS}(c)\vec{F}(c) \quad (3)$$

where $P_{RS}(c)$ is the likelihood of the category *c*. Finally, the deep similarity between a SAMT-style syntactic category *tc* of a test source string and a fuzzy-tree to exact-tree rule is computed as follows:

$$DeepSim(tc, RS)=\vec{F}(tc)\cdot\vec{F}(RS) \quad (4)$$

This deep similarity score will serve as a useful feature in the string-to-tree model which will enable the model to learn how to take account of the source-side syntax during translation.

We have ignored the details of latent syntactic category induction in this paper. In brief, the set of latent syntactic categories is automatically induced from a source-side parsed, word-aligned parallel corpus. The EM algorithm is employed to induce the parameters. We simply follow the algorithm of (Huang et al., 2010), except that we replace the tag sequence with SAMT-style syntactic categories.

4 Rule Binarization

In the baseline string-to-tree model, the rules are not in Chomsky Normal Form. There are several ways to ensure cubic-time decoding. One way is to prune the extracted rules using a scope-3 grammar and do SCFG decoding without binarization (Hopkins and Lengmead, 2010). The other, and most popular way is to binarize the translation rules (Zhang et al., 2006). We adopt the latter approach for efficient decoding with integrated *n*-gram language models since this binarization technique has been well studied in string-to-tree

translation. However, when the rules' source string is decorated with syntax (fuzzy-tree to exact-tree rules), how should we binarize these rules?

We use the rule r_n in Figure 2 for illustration:

$r_n:\ x_2\ x_0\ x_1\ \{PP*VP\} \rightarrow VP(x_0:VB\ \ x_1:NP\ \ x_2:PP)$.

Without regarding the source-side syntax, we obtain the following two binarized rules:

$$B1:\quad x_2\ x_{0*1} \rightarrow VP\left(x_{0*1}:V_{x_0*x_1}\ \ x_2:PP\right)$$

$$B2:\quad x_0\ x_1 \rightarrow V_{x_0*x_1}\left(x_0:VB\ \ x_1:NP\right)$$

Since the source-side syntax *PP*VP* in rule r_n only accounts for the entire source side, it is unclear how to annotate the source side of a partial rule such as the second binary rule *B2*.

Analyzing the derivation process, we observe that a partial rule such as binary rule *B2* never appears in the final derivation unless the rooted binary rule *B1* also appears in the derivation. Based on this observation, we design a heuristic[3] strategy: we simply attach the syntax *PP*VP* in the rooted binary rule *B1*, and do not decorate other binary rules with source syntax. Thus rule r_n will be binarized as:

$$(1)\quad x_2\ x_{0*1}\ \{PP*VP\} \rightarrow VP\left(x_{0*1}:V_{x_0*x_1}\ \ x_2:PP\right)$$

$$(2)\quad x_0\ x_1 \rightarrow V_{x_0*x_1}\left(x_0:VB\ \ x_1:NP\right)$$

5 Translation Model and Decoding

The proposed translation system is an augmentation of the string-to-tree model. In the baseline string-to-tree model, the decoder searches for the optimal derivation d^* that parses a source string f into a target tree e_t from all possible derivations D:

$$d^* = \arg\max_{d\in D} \lambda_1 \log p_{LM}\left(\tau(d)\right) + \lambda_2\left|\tau(d)\right| + \lambda_3|d| + R(d\mid f) \quad (5)$$

where the first element is a language model score in which $\tau(d)$ is the target yield of derivation d; the second element is the translation length penalty; the third element is used to control the derivation length; and the last element is a translation score that includes six features:

$$\begin{aligned} R(d\mid f) = &\sum_{r\in d} \lambda_4 \log p\left(r \mid root(r)\right) + \lambda_5 \log p\left(r \mid lhs(r)\right) \\ &+ \lambda_6 \log p\left(r \mid rhs(r)\right) + \lambda_7 \log p_{lex}\left(lhs(r) \mid rhs(r)\right) \\ &+ \lambda_8 \log p_{lex}\left(rhs(r) \mid lhs(r)\right) + \lambda_9 \delta\left(is\_comp\right) \end{aligned} \quad (6)$$

In equation (6), the first three elements denote the conditional probability of the rule given the root, the source-hand side, and the target-hand side. The next two elements are bidirectional lexical translation probabilities. The last element is the preferred binary feature for learning: either the composed rule or the minimal rule.

In our source-syntax-augmented model, the decoder also searches for the best derivation. With the help of the source syntactic information, the derivation rules in our new model are much more distinguishable than that in the string-to-tree model:

$$d^* = \arg\max_{d\in D} \lambda_1 \log p_{LM}\left(\tau(d)\right) + \lambda_2\left|\tau(d)\right| + \lambda_3|d| + \overline{R}(d\mid f) \quad (7)$$

Here, all elements except the last one are the same as in the string-to-tree model. The last item is:

$$\begin{aligned} \overline{R}(d\mid f) = &R(d\mid f) \\ &+ \sum_{r\in d} \delta(DeepSim)\lambda_{10} \log\left(DeepSim(tag, r)\right) \\ &+ \delta(likelihood)\lambda_{11} \log\left(likelihood(tag, r)\right) \\ &+ \delta(01)\left\{\lambda_{12}\delta(match) + \lambda_{13}\delta(unmatch)\right\} \end{aligned} \quad (8)$$

The 0-1 matching[4] is triggered only when we set $\delta(01)=1$. The other two fuzzy matching algorithms are triggered in a similar way.

During decoding, we use a CKY-style parser with beam search and cube-pruning (Huang and Chiang, 2007) to decode the new source sentences.

6 Experiments

6.1 Experimental Setup

The experiments are conducted on Chinese-to-English translation, with training data consisting of about 19 million English words and 17 million Chinese words[5]. We performed bidirectional word alignment using GIZA++, and employed the *grow-diag-final* balancing strategy to generate the final

[3] We call it heuristic because there may be other syntactic annotation strategies for the binarized rules. It should be noted that our strategy makes the annotated binarized rules equivalent to the original rule.

[4] In theory, the features *unmatch_count*, *match_count* and *derivation_length* are linearly dependent, so the *unmatch_count* is redundant. In practice, since the derivation may include glue rules which are not scored by fuzzy matching. Thus, "*unmatch_count* + *match_count* + *glue_rule_number* = *derivation_length*".

[5] LDC catalog number: LDC2002E18, LDC2003E14, LDC2003E07, LDC2004T07 and LDC2005T06.

symmetric word alignment. We parsed both sides of the parallel text with the Berkeley parser (Petrov et al., 2006) and trained a 5-gram language model with the target part of the bilingual data and the Xinhua portion of the English Gigaword corpus.

For tuning and testing, we use NIST MT evaluation data for Chinese-to-English from 2003 to 2006 (MT03 to MT06). The development data set comes from MT06 in which sentences with more than 20 words are removed to speed up MERT[6] (Och, 2003). The test set includes MT03 to MT05.

We implemented the baseline string-to-tree system ourselves according to (Galley et al., 2006; Marcu et al., 2006). We extracted minimal GHKM rules and the rules of SPMT Model 1 with source language phrases up to length *L=4*. We further extracted composed rules by composing two or three minimal GHKM rules. We also ran the state-of-the-art hierarchical phrase-based system Joshua (Li et al., 2009) for comparison. In all systems, we set the beam size to 200. The final translation quality is evaluated in terms of case-insensitive BLEU-4 with shortest length penalty. The statistical significance test is performed using the re-sampling approach (Koehn, 2004).

6.2 Results

Table 1 shows the translation results on development and test sets. First, we investigate the performance of the strong baseline string-to-tree model (*s2t* for short). As the table shows, *s2t* outperforms the hierarchical phrase-based system *Joshua* by more than 1.0 BLEU point in all translation tasks. This result verifies the superiority of the baseline string-to-tree model.

With the *s2t* system providing a baseline, we further study the effectiveness of our source-syntax-augmented string-to-tree system with fuzzy-tree to exact-tree rules (we use *FT2ET* to denote our proposed system). The last three lines in Table 1 show that, for each fuzzy matching algorithm, our new system *TF2ET* performs significantly better than the baseline *s2t* system, with an improvement of more than 0.5 absolute BLEU points in all tasks. This result demonstrates the success of our new method of incorporating source-side syntax in a string-to-tree model.

[6] The average decoding speed is about 50 words per minute in the baseline string-to-tree system and our proposed systems.

System		MT06 (dev)	MT03	MT04	MT05
Joshua		29.42	28.62	31.52	31.39
s2t		30.84	29.75	32.68	32.41
FT2ET	*0-1*	31.61**	30.60**	33.45**	33.37**
	LH	31.35*	30.34*	33.21*	33.05*
	DeepSim	31.77**	30.82**	33.69**	33.50**

Table 1: Results (in BLEU scores) of different translation models in multiple tasks. *LH*=likelihood. *or**=significantly better than *s2t* system (p<0.05 or 0.01 respectively).

	Very similar $\vec{F}(c)\cdot\vec{F}(c')$>0.9	Very dissimilar $\vec{F}(c)\cdot\vec{F}(c')$<0.1
ADJP	JJ; AD\ADJP	VP; ADVP\NP
NP	DT*NN; LCP*P*NP	CP; BA*CP

Table 2: Example of similar and dissimilar categories.

Specifically, the *FT2ET* system with *deep similarity matching* obtains the best translation quality in all tasks and surpasses the baseline *s2t* system by 0.93 BLEU points in development data and by more than 1.0 BLEU point in test sets. The *0-1 matching* algorithm is simple but effective, and it yields quite good performance (line 3). The contribution of *0-1 matching* as reflected in our experiments is consistent with the conclusions of (Marton and Resnik, 2008; Chiang, 2010). By contrast, the system with *likelihood matching* does not perform as well as the other two algorithms, although it also significantly improves the baseline *s2t* in all tasks.

6.3 Analysis and Discussion

We are a bit surprised at the large improvement gained by the 0-1 matching algorithm. This algorithm has several advantages: it is simple and easy to implement, and enhances the translation model by enabling its rules to take account of the source-side syntax to some degree. However, a major deficiency of this algorithm is that it does not make full use of the source side syntax, since it retains only the most frequent SAMT-style syntactic category to describe the rule's source syntax. Thus this algorithm penalizes all the other categories equally, although some may be more frequent than others, as in the case of *DEG* and *DEV* in the rule 的$\{DEC:11233, DEG:11073, DEV:65\}\rightarrow IN(of)$.

To some extent, the likelihood matching algorithm solves the main problem of 0-1 matching.

Instead of rewarding or penalizing, this algorithm uses the likelihood of the syntactic category to approximate the degree of matching between the test source syntactic category and the rule. For a category not in the rule's source syntactic category set, the likelihood algorithm computes a smoothed likelihood. However, the likelihood algorithm does not in fact lead to very promising improvement. We conjecture that this disappointing performance is due to the simple smoothing method we employed. Future work will investigate more fully.

Compared with the above two matching algorithms, the deep similarity matching algorithm based on latent syntactic distribution is much more beautiful in theory. This algorithm can successfully measure the similarity between any two SAMT-style syntactic categories (Table 2 gives some examples of similar and dissimilar category pairs). Then it can accurately compute the degree of matching between a test source syntactic category and a fuzzy-tree to exact-tree rule. Thus this algorithm obtains the best translation quality. However, the deep similarity matching algorithm has two practical shortcomings. First, it is not easy to determine the number of latent categories. We have to conduct multiple experiments to arrive at a number which can yield a tradeoff between translation quality and model complexity. In our work, we have tried the numbers *n=4, 8, 16, 32,* and have found *n=16* to give the best tradeoff. The second shortcoming is that the induction of latent syntactic categories has been very time consuming, since we have applied the EM algorithm to the entire source-parsed parallel corpus. Even with *n=8*, it took more than a week to induce the latent syntactic categories on our middle-scale training data when using a Xeon four-core computer ($2.5GHz \times 2CPU \times 16GB\ memory$). When the training data contains tens of millions of sentence pairs, the computation time may no longer be tolerable.

Table 3 shows some translation examples for comparison. In the first example, the Chinese preposition word *和* is mistakenly translated into English conjunction word *and* in *Joshua* and baseline string-to-tree system *s2t*, however, our source-syntax-augmented system *FT2ET-DeepSim* correctly converts the Chinese word *和* into English preposition *with* and finally yield the right translation. In the second example, our proposed system moves the prepositional phrase *at an early date* after the sibling verb phrase. It is more reasonable compared with the baseline system *s2t*. In the third example, the proposed system *FT2ET-DeepSim* successfully recognizes the Chinese long prepositional phrase *在 与 中国 总理 温家宝 举行 峰会 后 发布 的联合 声明 中* and short verb phrase *说*, and obtains the correct phrase reordering at last.

7 Related Work

Several studies have tried to incorporate source or target syntax into translation models in a fuzzy manner.

Zollmann and Venugopal (2006) augment the hierarchical string-to-string rules (Chiang, 2005) with target-side syntax. They annotate the target side of each string-to-string rule using SAMT-style syntactic categories and aim to generate the output more syntactically. Zhang et al. (2010) base their approach on tree-to-string models, and generate grammatical output more reliably with the help of tree-to-tree sequence rules. Neither of them builds target syntactic trees using target syntax, however. Thus they can be viewed as integrating target syntax in a fuzzy manner. By contrast, we base our approach on a string-to-tree model which does construct target syntactic trees during decoding.

(Marton and Resnik, 2008; Chiang et al., 2009 and Huang et al., 2010) apply fuzzy techniques for integrating source syntax into hierarchical phrase-based systems (Chiang, 2005, 2007). The first two studies employ 0-1 matching and the last tries deep similarity matching between two tag sequences. By contrast, we incorporate source syntax into a string-to-tree model. Furthermore, we apply fuzzy syntactic annotation on each rule's source string and design three fuzzy rule matching algorithms.

Chiang (2010) proposes a method for learning to translate with both source and target syntax in the framework of a hierarchical phrase-based system. He not only executes 0-1 matching on both sides of rules, but also designs numerous features such as $root_{X,X'}$ which counts the number of rules whose source-side root label is X and target-side root label is X'. This fuzzy use of source and target syntax enables the translation system to learn which tree labels are similar enough to be compatible, which ones are harmful to combine, and which ones can be ignored. The differences between us are twofold: 1) his work applies fuzzy syntax in both sides, while ours bases on the string-

1	*Source sentence*	海珊 也 [和 恐怖 组织网] 建立 了 联系
	Reference	hussein also established ties with terrorist networks
	Joshua	hussein also and terrorist networks established relations
	s2t	hussein also and terrorist networks established relations
	FT2ET- DeepSim	hussein also established relations with terrorist networks
2	*Source sentence*	… [以 期] [早日] [结束] [以 巴 之间 多年 的 流血 冲突]
	Reference	.. to end years of bloody conflict between israel and palestine as soon as possible .. to end at an early date years of bloody conflict between israel and palestine
	Joshua	… in the early period to end years of blood conflict between israel and palestine
	s2t	… at an early date to end years of blood conflict between israel and palestine
	FT2ET- DeepSim	… to end years of blood conflict between israel and palestine at an early date
3	*Source sentence*	欧盟 [在 与 中国 总理 温家宝 举行 峰会 后 发布 的联合 声明 中] [说] …
	Reference	the europen union said in a joint statement issued after its summit meeting with china 's premier wen jiabao … in a joint statement released after the summit with chinese premier wen jiabao , the europen union said …
	Joshua	the europen union with chinese premier wen jiabao in a joint statement issued after the summit meeting said …
	s2t	the europen union in a joint statement issued after the summit meeting with chinese premier wen jiabao said …
	FT2ET- DeepSim	the europen union said in a joint statement issued after the summit meeting with chinese premier wen jiabao …

Table 3: Some translation examples produced by *Joshua*, string-to-tree system *s2t* and source-syntax-augmented string-to-tree system *FT2ET* with deep similarity matching algorithm

to-tree model and applies fuzzy syntax on source side; and 2) we not only adopt the 0-1 fuzzy rule matching algorithm, but also investigate likelihood matching and deep similarity matching algorithms.

8 Conclusion and Future Work

In this paper, we have proposed a new method for augmenting string-to-tree translation models with fuzzy use of the source syntax. We first applied a fuzzy annotation method which labels the source side of each string-to-tree rule with SAMT-style syntactic categories. Then we designed and explored three fuzzy rule matching algorithms: 0-1 matching, likelihood matching, and deep similarity matching. The experiments show that our new system significantly outperforms the strong baseline string-to-tree system. This substantial improvement verifies that our fuzzy use of source syntax is effective and can enhance the ability to choose proper translation rules during decoding while guaranteeing grammatical output with explicit target trees. We believe that our work may demonstrate effective ways of incorporating both-side syntax in a translation model to yield promising results.

Next, we plan to further study the likelihood fuzzy matching and deep similarity matching algorithms in order to fully exploit their potential. For example, we will combine the merits of 0-1 matching and likelihood matching so as to avoid the setting of parameter *m* in likelihood matching. We also plan to explore another direction: we will annotate the source side of each string-to-tree rule with subtrees or subtree sequences. We can then apply tree-kernel methods to compute a degree of matching between a rule and a test source subtree or subtree sequence.

Acknowledgments

The research work has been funded by the Natural Science Foundation of China under Grant No. 60975053, 61003160 and 60736014 and supported by the External Cooperation Program of the Chinese Academy of Sciences. We would also like to thank Mark Seligman and Yu Zhou for revising the early draft, and anonymous reviewers for their valuable suggestions.

References

Yehoshua Bar-Hillel, 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29 (1). pages 47-58.

David Chiang, 2005. A hiearchical phrase-based model for statistical machine translation. In *Proc. of ACL* 2005, pages 263-270.

David Chiang, 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33 (2). pages 201-228.

David Chiang, 2010. Learning to translate with source and target syntax. In *Proc. of ACL 2010*, pages 1443-1452.

David Chiang, Kevin Knight and Wei Wang, 2009. 11,001 new features for statistical machine translation. In *Proc. of NAACL 2009*, pages 218-226.

Brooke Cowan, Ivona Kucerova and Michael Collins, 2006. A discriminative model for tree-to-tree translation. In *Proc. of EMNLP*, pages 232-241.

Yuan Ding and Martha Palmer, 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of ACL 2005*, pages 541-548.

Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu, 2004. What's in a translation rule. In *Proc. of HLT-NAACL 2004*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer, 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of ACL-COLING 2006*.

Mark Hopkins and Greg Langmead, 2010. SCFG decoding without binarization. In *Proc. of EMNLP 2010*, pages 646-655.

Liang Huang and David Chiang, 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL 2007*, pages 144-151.

Liang Huang, Kevin Knight and Aravind Joshi, 2006. A syntax-directed translator with extended domain of locality. In *Proc. of AMTA 2006*, pages 65-73.

Zhongqiang Huang, Martin Cmejrek and Bowen Zhou, 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proc. of EMNLP 2010*, pages 138-147.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst, 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL 2007*, pages 177-180.

Philipp Koehn, 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP 2004*, pages 388–395.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N.G. Thornton, Jonathan Weese and Omar F. Zaidan, 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proc. of ACL 2009*, pages 135-139.

Yang Liu, Qun Liu and Shouxun Lin, 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL-COLING 2006*, pages 609-616.

Yang Liu, Yajuan Lv and Qun Liu, 2009. Improving tree-to-tree translation with packed forests. In *Proc. of ACL-IJCNLP 2009*, pages 558-566.

Daniel Marcu, Wei Wang, Abdessamad Echihabi and Kevin Knight, 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP 2006*, pages 44-52.

Yuval Marton and Philip Resnik, 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proc. of ACL-08: HLT*. pages 1003–1011.

Haitao Mi, Liang Huang and Qun Liu, 2008. Forest-based translation. In *Proc. of ACL-08: HLT*. pages 192–199.

Haitao Mi and Qun Liu, 2010. Constituency to dependency translation with forests. In *Proc. of ACL 2010*, pages 1433-1442.

Tom M. Mitchell, 1997. Machine learning. *Mac Graw Hill*.

Franz Josef Och, 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*, pages 160-167.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein, 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL 2006*, pages 433-440.

Chris Quirk, Arul Menezes and Colin Cherry, 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL 2005,* pages 271-279.

Libin Shen, Jinxi Xu and Ralph Weischedel, 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL-08: HLT*, pages 577-585.

Tong Xiao, Jingbo Zhu, Muhua Zhu and and Huizhen Wang, 2010. Boosting-based System Combination for Machine Translation. In *Proc. of ACL 2010*, pages 739-748.

Hao Zhang, Liang Huang, Daniel Gildea and Kevin Knight, 2006. Synchronous binarization for machine translation. In *Proc. of HLT-NAACL 2006*, pages 256-263.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, Chew Lim Tan, 2009. Forest-based tree sequence to string translation model. In *Proc. of ACL-IJCNLP 2009*, pages 172-180.

Hui Zhang, Min Zhang, Haizhou Li and Chng Eng Siong, 2010. Non-isomorphic forest pair translation. In *Proc. of EMNLP 2010*, pages 440-450.

Andreas Zollmann and Ashish Venugopal, 2006. Syntax augmented machine translation via chart parsing. In *Proc. of Workshop on Statistical Machine Translation 2006*, pages 138-141.

A Novel Dependency-to-String Model for Statistical Machine Translation

Jun Xie, Haitao Mi and Qun Liu
Key Laboratory of Intelligent Information Processiong
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{junxie,htmi,liuqun}@ict.ac.cn

Abstract

Dependency structure, as a first step towards semantics, is believed to be helpful to improve translation quality. However, previous works on dependency structure based models typically resort to insertion operations to complete translations, which make it difficult to specify ordering information in translation rules. In our model of this paper, we handle this problem by directly specifying the ordering information in head-dependents rules which represent the source side as head-dependents relations and the target side as strings. The head-dependents rules require only substitution operation, thus our model requires no heuristics or separate ordering models of the previous works to control the word order of translations. Large-scale experiments show that our model performs well on long distance reordering, and outperforms the state-of-the-art constituency-to-string model (+1.47 BLEU on average) and hierarchical phrase-based model (+0.46 BLEU on average) on two Chinese-English NIST test sets without resort to phrases or parse forest. For the first time, a source dependency structure based model catches up with and surpasses the state-of-the-art translation models.

1 Introduction

Dependency structure represents the grammatical relations that hold between the words in a sentence. It encodes semantic relations directly, and has the best inter-lingual phrasal cohesion properties (Fox, 2002). Those attractive characteristics make it possible to improve translation quality by using dependency structures.

Some researchers pay more attention to use dependency structure on the target side. (Shen et al., 2008) presents a string-to-dependency model, which restricts the target side of each hierarchical rule to be a well-formed dependency tree fragment, and employs a dependency language model to make the output more grammatically. This model significantly outperforms the state-of-the-art hierarchical phrase-based model (Chiang, 2005). However, those string-to-tree systems run slowly in cubic time (Huang et al., 2006).

Using dependency structure on the source side is also a promising way, as tree-based systems run much faster (linear time vs. cubic time, see (Huang et al., 2006)). Conventional dependency structure based models (Lin, 2004; Quirk et al., 2005; Ding and Palmer, 2005; Xiong et al., 2007) typically employ both substitution and insertion operation to complete translations, which make it difficult to specify ordering information directly in the translation rules. As a result, they have to resort to either heuristics (Lin, 2004; Xiong et al., 2007) or separate ordering models (Quirk et al., 2005; Ding and Palmer, 2005) to control the word order of translations.

In this paper, we handle this problem by directly specifying the ordering information in head-dependents rules that represent the source side as head-dependents relations and the target side as string. The head-dependents rules have only one substitution operation, thus we don't face the problems appeared in previous work and get rid of the

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 216–226,
Edinburgh, Scotland, UK, July 27–31, 2011.

heuristics and ordering model. To alleviate data sparseness problem, we generalize the lexicalized words in head-dependents relations with their corresponding categories.

In the following parts, we first describe the motivation of using head-dependents relations (Section 2). Then we formalize our grammar (Section 3), present our rule acquisition algorithm (Section 4), our model (Section 5) and decoding algorithm (Section 6). Finally, large-scale experiments (Section 7) show that our model exhibits good performance on long distance reordering, and outperforms the state-of-the-art tree-to-string model (+1.47 BLEU on average) and hierarchical phrase-based model (+0.46 BLEU on average) on two Chinese-English NIST test sets. For the first time, a source dependency tree based model catches up with and surpasses the state-of-the-art translation models.

2 Dependency Structure and Head-Dependents Relation

2.1 Dependency Sturcture

A dependency structure for a sentence is a directed acyclic graph with words as nodes and modification relations as edges. Each edge direct from a head to a dependent. Figure 1 (a) shows an example dependency structure of a Chinese sentence.

2010年 FIFA 世界杯 在 南非 成功 举行

2010 FIFA [World Cup] in/at [South Africa] successfully hold

Each node is annotated with the part-of-speech (POS) of the related word.

For convenience, we use the lexicon dependency grammar (Hellwig, 2006) which adopts a bracket representation to express a projective dependency structure. The dependency structure of Figure 1 (a) can be expressed as:

((2010年) (FIFA) 世界杯) (在(南非)) (成功) 举行

where the lexicon in brackets represents the dependents, while the lexicon out the brackets is the head.

To construct the dependency structure of a sentence, the most important thing is to establish dependency relations and distinguish the head from the dependent. Here are some criteria (Zwicky, 1985; Hudson, 1990) for identifying a syntactic relation between a *head* and a *dependent* between a head-dependent pair:

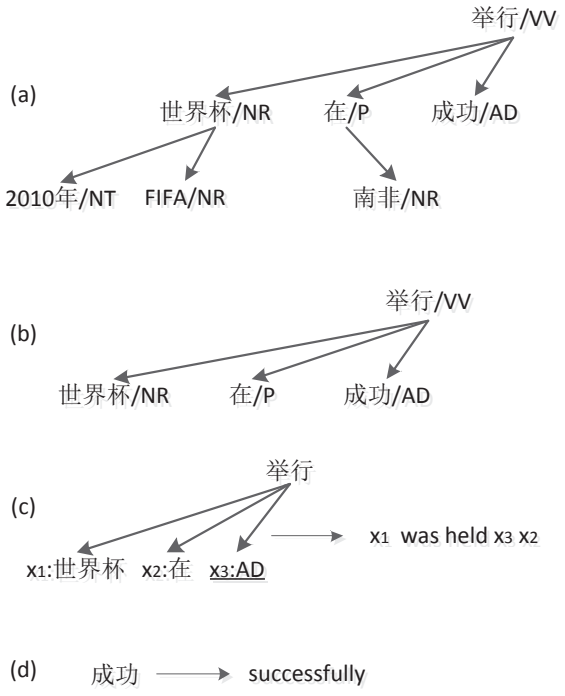


Figure 1: Examples of dependency structure (a), head-dependents relation (b), head-dependents rule (r_1 of Figure 2) and head rule (d). Where "x_1:世界杯" and "x_2:在" indicate substitution sites which can be replaced by a subtree rooted at "世界杯" and "在" respectively. "x_3:AD"indicates a substitution site that can be replaced by a subtree whose root has part-of-speech "AD". The underline denotes a leaf node.

1. *head* determines the syntactic category of *C*, and can often replace *C*;
2. *head* determines the semantic category of *C*; *dependent* gives semantic specification.

2.2 Head-Dependents Relation

A head-dependents relation is composed of a head and all its dependents as shown in Figure 1(b).

Since all the head-dependent pairs satisfy criteria 1 and 2, we can deduce that a head-dependents relation *L* holds the property that *the head determines the syntactic and semantic categories of L, and can often replace L.* Therefore, we can recur-

sively replace the bottom level head-dependent relations of a dependency structure with their heads until the root. This implies an representation of the generation of a dependency structure on the basis of head-dependents relation.

Inspired by this, we represent the translation rules of our dependency-to-string model on the foundation of head-dependents relations.

3 Dependency-to-String Grammar

Figure 1 (c) and (d) show two examples of the translation rules used in our dependency-to-string model. The former is an example of *head-dependent rules* that represent the source side as head-dependents relations and act as both translation rules and reordering rules. The latter is an example of *head rules* which are used for translating words.

Formally, a dependency-to-string grammar is defined as a tuple $\langle \Sigma, N, \Delta, R \rangle$, where Σ is a set of source language terminals, N is a set of categories for the terminals in Σ , Δ is a set of target language terminals, and R is a set of translation rules. A rule r in R is a tuple $\langle t, s, \phi \rangle$, where:

- t is a node labeled by terminal from Σ; or a head-dependents relation of the source dependency structures, with each node labeled by a terminal from Σ or a variable from a set $X = \{x_1, x_2, ...\}$ constrained by a terminal from Σ or a category from N;
- $s \in (X \cup \Delta)^*$ is the target side string;
- ϕ is a one-to-one mapping from nonterminals in t to variables in s.

For example, the head-dependents rule shown in Figure 1 (c) can be formalized as:

t = ((x_1:世界杯) (x_2:在) ($\underline{x_3}$:AD) 举行)
s = x_1 was held x_3 x_2
ϕ = {x_1:世界杯 $\leftrightarrow$ x_1, x_2:在 $\leftrightarrow$ x_2, x_3:AD $\leftrightarrow$ x_3}

where the underline indicates a leaf node, and x_i:*letters* indicates a pair of variable and its constraint.

A derivation is informally defined as a sequence of steps converting a source dependency structure into a target language string, with each step applying one translation rule. As an example, Figure 2 shows the derivation for translating a Chinese (CH) sentence into an English (EN) string.

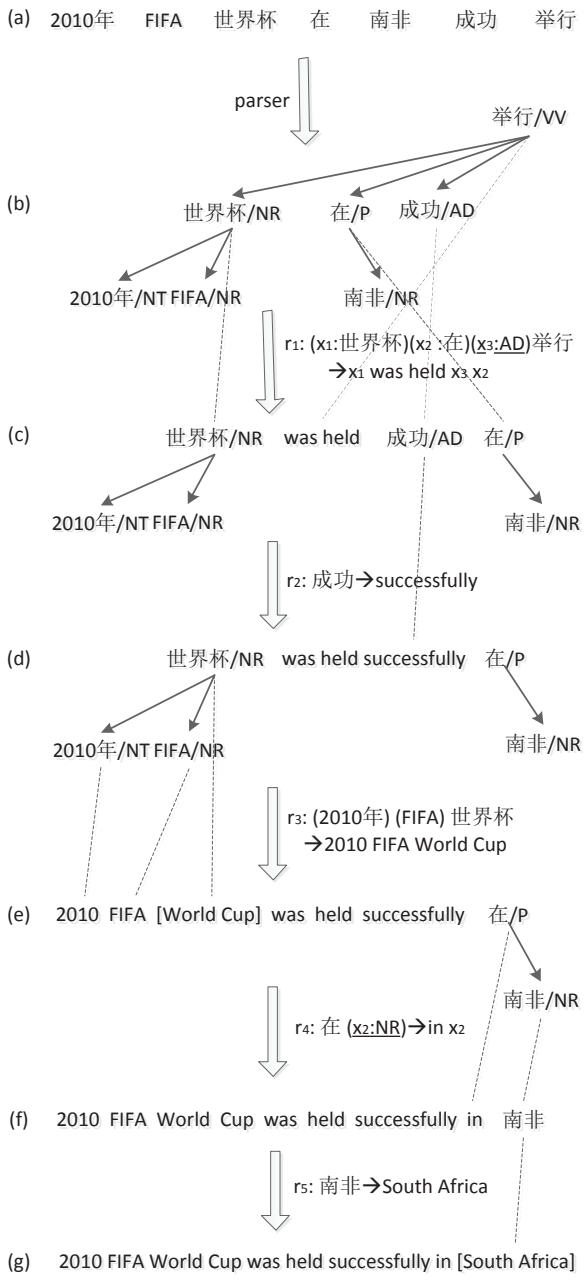


Figure 2: An example derivation of dependency-to-string translation. The dash lines indicate the reordering when employing a head-dependents rule.

CH 2010年 FIFA 世界杯 在 南非 成功 举行

EN *2010 FIFA World Cup was held successfully in South Africa*

The Chinese sentence (a) is first parsed into a dependency structure (b), which is converted into an English string in five steps. First, at the root node, we apply head-dependents rule r_1 shown in Figure 1(c) to translate the top level head-dependents relation and result in three unfinished substructures and target string in (c). The rule is particular interesting since it captures the fact: in Chinese prepositional phrases and adverbs typically modify verbs on the left, whereas in English prepositional phrases and adverbs typically modify verbs on the right. Second, we use head rule r_2 translating "成功" into "successfully" and reach situation (d). Third, we apply head-dependents rule r_3 translating the head-dependents relation rooted at "世界杯" and yield (e). Fourth, head-dependents rules r_5 partially translate the subtree rooted at "在" and arrive situation in (f). Finally, we apply head rule r_5 translating the residual node "南非" and obtain the final translation in (g).

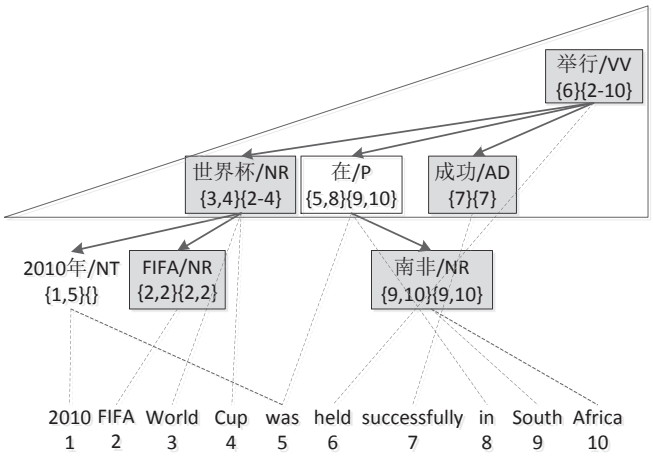


Figure 3: An annotated dependency structure. Each node is annotated with two spans, the former is head span and the latter dependency span. The nodes in *acceptable head set* are displayed in gray, and the nodes in *acceptable dependent set* are denoted by boxes. The triangle denotes the only acceptable head-dependents fragment.

4 Rule Acquisition

The rule acquisition begins with a word-aligned corpus: a set of triples $\langle T, S, A\rangle$, where T is a source dependency structure, S is a target side sentence, and A is an alignment relation between T and S. We extract from each triple $\langle T, S, A\rangle$ head rules that are consistent with the word alignments and head-dependents rules that satisfy the intuition that syntactically close items tend to stay close across languages. We accomplish the rule acquisition through three steps: tree annotation, head-dependents fragments identification and rule induction.

4.1 Tree Annotation

Given a triple $\langle T, S, A\rangle$ as shown in Figure 3, we first annotate each node n of T with two attributes: head span and dependency span, which are defined as follows.

Definition 1. *Given a node n, its **head span** $hsp(n)$ is a set of index of the target words aligned to n.*

For example, hsp(2010年)=$\{1, 5\}$, which corresponds to the target words "*2010*" and "*was*".

Definition 2. *A head span $hsp(n)$ is **consistent** if it satisfies the following property:*

$$\forall_{n' \neq n} hsp(n') \cap hsp(n) = \varnothing.$$

For example, hsp(南非) is consistent, while hsp(2010年) is not consistent since hsp(2010年) $\cap$ hsp(在) $= 5$.

Definition 3. *Given a head span $hsp(n)$, its **closure** cloz(hsp(n)) is the smallest contiguous head span that is a superset of $hsp(n)$.*

For example, $cloz(hsp$(2010年)) = $\{1, 2, 3, 4, 5\}$, which corresponds to the target side word sequence "*2010 FIFA World Cup was*". For simplicity, we use $\{$1-5$\}$ to denotes the contiguous span $\{1, 2, 3, 4, 5\}$.

Definition 4. *Given a subtree T' rooted at n, the **dependency span** $dsp(n)$ of n is defined as:*

$$dsp(n) = cloz(\bigcup_{\substack{n' \in T' \\ hsp(n') \text{ is consistent}}} hsp(n')).$$

If the head spans of all the nodes of T' is not consistent, $dsp(n) = \varnothing$.

For example, since hsp(在) is not consistent, dsp(在)=dsp(南非)=$\{9, 10\}$, which corresponds to the target words "*South*" and "*Africa*".

The tree annotation can be accomplished by a single postorder transversal of T. The extraction of *head rules* from each node can be readily achieved with the same criteria as (Och and Ney, 2004). In

the following, we focus on *head-dependents rules* acquisition.

4.2 Head-Dependents Fragments Identification

We then identify the head-dependents fragments that are suitable for rule induction from the annotated dependency structure.

To facilitate the identification process, we first define two sets of dependency structure related to head spans and dependency spans.

Definition 5. *A **acceptable head set** ahs(T) of a dependency structure T is a set of nodes, each of which has a consistent head span.*

For example, the elements of the acceptable head set of the dependency structure in Figure 3 are displayed in gray.

Definition 6. *A **acceptable dependent set** adt(T) of a dependency structure T is a set of nodes, each of which satisfies:* $dep(n) \neq \varnothing$.

For example, the elements of the acceptable dependent set of the dependency structure in Figure 3 are denoted by boxes.

Definition 7. *We say a head-dependents fragments is **acceptable** if it satisfies the following properties:*

1. *the root falls into acceptable head set;*
2. *all the sinks fall into acceptable dependent set.*

An acceptable head-dependents fragment holds the property that the head span of the root and the dependency spans of the sinks do not overlap with each other, which enables us to determine the reordering in the target side.

The identification of *acceptable* head-dependents fragments can be achieved by a single preorder transversal of the annotated dependency structure. For each accessed internal node n, we check whether the head-dependents fragment f rooted at n is acceptable. If f is acceptable, we output an acceptable head-dependents fragment; otherwise we access the next node.

Typically, each acceptable head-dependents fragment has three types of nodes: *internal nodes*, internal nodes of the dependency structure; *leaf nodes*, leaf nodes of the dependency structure; *head node*, a special internal node acting as the head of the related head-dependents relation.

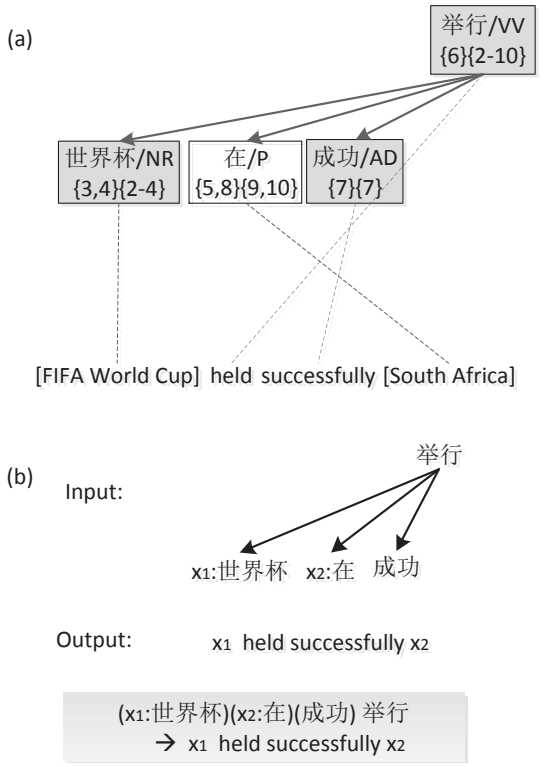


Figure 4: A lexicalized head-dependents rule (b) induced from the only acceptable head-dependents fragment (a) of Figure 3.

4.3 Rule Induction

From each acceptable head-dependents fragment, we induce a set of lexicalized and unlexicalized head-dependents rules.

4.3.1 Lexicalized Rule

We induce a lexicalized head-dependents rule from an acceptable head-dependents fragment by the following procedure:

1. extract the head-dependents relation and mark the internal nodes as substitution sites. This forms the input of a head-dependents rule;
2. place the nodes in order according to the head span of the root and the dependency spans of the sinks, then replace the internal nodes with variables and the other nodes with the target words covered by their head spans. This forms the output of a head-dependents rule.

Figure 4 shows an acceptable head-dependents fragment and a lexicalized head-dependents rule in-

duced from it.

4.3.2 Unlexicalized Rules

Since head-dependents relations with verbs as heads typically consist of more than four nodes, employing only lexicalized head-dependents rules will result in severe sparseness problem. To alleviate this problem, we generalize the lexicalized head-dependents rules and induce rules with unlexicalized nodes.

As we know, the modification relation of a head-dependents relation is determined by the edges. Therefore, we can replace the lexical word of each node with its categories (i.e. POS) and obtain new head-dependents relations with unlexicalized nodes holding the same modification relation. Here we call the lexicalized and unlexicalized head-dependents relations as instances of the modification relation. For a head-dependents relation with m node, we can produce $2^m - 1$ instances with unlexicalized nodes. Each instance represents the modification relation with a different specification.

Based on this observation, from each lexicalized head-dependent rule, we generate new head-dependents rules with unlexicalized nodes according to the following principles:

1. change the aligned part of the target string into a new variable when turning a head node or a leaf node into its category;

2. keep the target side unchanged when turning a internal node into its category.

Restrictions: Since head-dependents relations with verbs as heads typically consists of more than four nodes, enumerating all the instances will result in a massive grammar with too many kinds of rules and inflexibility in decoding. To alleviate these problems, we filter the grammar with the following principles:

1. nodes of the same type turn into their categories simultaneously.

2. as for leaf nodes, only those with open class words can be turned into their categories. In our experiments of this paper, we only turn those dependents with POS tag in the set of {CD,DT,OD,JJ,NN,NR,NT,AD,FW,PN} into their categories.

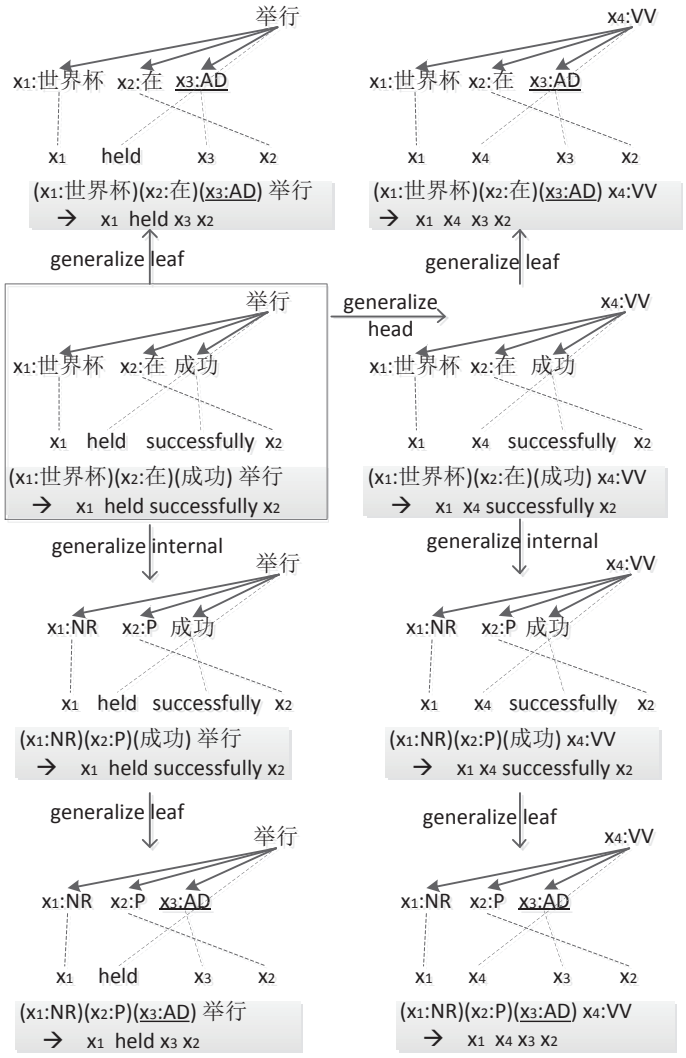


Figure 5: An illustration of rule generalization. Where "x_1:世界杯" and "x_2:在" indicate substitution sites which can be replaced by a subtree rooted at "世界杯" and "在" respectively. "x_3:AD"indicates a substitution site that can be replaced by a subtree whose root has part-of-speech "AD". The underline denotes a leaf node. The box indicates the starting lexicalized head-dependents rule.

Figure 5 illustrates the rule generalization process under these restrictions.

4.3.3 Unaligned Words

We handle the unaligned words of the target side by extending the head spans of the lexicalized head and leaf nodes on both left and right directions. This procedure is similar with the method of (Och and Ney, 2004) except that we might extend several

Algorithm 1: Algorithm for Rule Acquisition

Input: Source dependency structure T, target string S, alignment A
Output: Translation rule set R

```
HSet ← ACCEPTABLE_HEAD(T,S,A)
DSet ← ACCEPTABLE_DEPENDENT(T,S,A)
for each node n ∈ HSet do
    extract head rules
    append the extracted rules to R
    if ∀n' ∈ child(n) n' ∈ DSet
    then
        obtain a head-dependent fragment f
        induce lexicalized and unlexicalized head-dependents rules from f
        append the induced rules to R
    end
end
```

spans simultaneously. In this process, we might obtain $m(m \geq 1)$ head-dependents rules from a head-dependent fragment in handling unaligned words. Each of these rules is assigned with a fractional count $1/m$.

4.4 Algorithm for Rule Acquisition

The rule acquisition is a three-step process, which is summarized in Algorithm 1.

We take the extracted rule set as observed data and make use of relative frequency estimator to obtain the translation probabilities $P(t|s)$ and $P(s|t)$.

5 The model

Following (Och and Ney, 2002), we adopt a general log-linear model. Let d be a derivation that convert a source dependency structure T into a target string e. The probability of d is defined as:

$$P(d) \propto \prod_i \phi_i(d)^{\lambda_i} \quad (1)$$

where ϕ_i are features defined on derivations and λ_i are feature weights. In our experiments of this paper, we used seven features as follows:

- translation probabilities $P(t|s)$ and $P(s|t)$;
- lexical translation probabilities $P_{lex}(t|s)$ and $P_{lex}(s|t)$;
- rule penalty $exp(-1)$;
- language model $P_{lm}(e)$;
- word penalty $exp(|e|)$.

6 Decoding

Our decoder is based on bottom up chart parsing. It finds the best derivation d^* that convert the input dependency structure into a target string among all possible derivations D:

$$d^* = argmax_{d \in D} P(D) \quad (2)$$

Given a source dependency structure T, the decoder transverses T in post-order. For each accessed internal node n, it enumerates all instances of the related modification relation of the head-dependents relation rooted at n, and checks the rule set for matched translation rules. If there is no matched rule, we construct a *pseudo translation rule* according to the word order of the head-dependents relation. For example, suppose that we can not find any translation rule about to "(2010年) (FIFA) 世界杯", we will construct a pseudo translation rule "(x_1:2010年) (x_2:FIFA) x_3:世界杯 $\rightarrow$ x_1 x_2 x_3". A larger translation is generated by substituting the variables in the target side of a translation rule with the translations of the corresponding dependents. We make use of cube pruning (Chiang, 2007; Huang and Chiang, 2007) to find the k-best items with integrated language model for each node.

To balance performance and speed, we prune the search space in several ways. First, beam thresh-

old β , items with a score worse than β times of the best score in the same cell will be discarded; second, beam size b, items with a score worse than the bth best item in the same cell will be discarded. The item consist of the necessary information used in decoding. Each cell contains all the items standing for the subtree rooted at it. For our experiments, we set $\beta = 10^{-3}$ and $b = 300$. Additionally, we also prune rules that have the same source side ($b = 100$).

7 Experiments

We evaluated the performance of our dependency-to-string model by comparison with replications of the hierarchical phrase-based model and the tree-to-string models on Chinese-English translation.

7.1 Data preparation

Our training corpus consists of 1.5M sentence pairs from LDC data, including LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

We parse the source sentences with Stanford Parser (Klein and Manning, 2003) into projective dependency structures, whose nodes are annotated by POS tags and edges by typed dependencies. In our implementation of this paper, we make use of the POS tags only.

We obtain the word alignments by running GIZA++ (Och and Ney, 2003) on the corpus in both directions and applying "grow-diag-and" refinement(Koehn et al., 2003).

We apply SRI Language Modeling Toolkit (Stolcke, 2002) to train a 4-gram language model with modified Kneser-Ney smoothing on the Xinhua portion of the Gigaword corpus.

We use NIST MT Evaluation test set 2002 as our development set, NIST MT Evaluation test set 2004 (MT04) and 2005 (MT05) as our test set. The quality of translations is evaluated by the *case insensitive* NIST BLEU-4 metric (Papineni et al., 2002).[1]

We make use of the standard MERT (Och, 2003) to tune the feature weights in order to maximize the system's BLEU score on the development set.

[1] ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

System	Rule #	MT04(%)	MT05(%)
cons2str	30M	34.55	31.94
hiero-re	148M	35.29	33.22
dep2str	56M	**35.82**$^{+}$	**33.62**$^{+}$

Table 1: Statistics of the extracted rules on training corpus and the BLEU scores on the test sets. Where "+" means *dep2str* significantly better than *cons2str* with $p < 0.01$.

7.2 The baseline models

We take a replication of Hiero (Chiang, 2007) as the hierarchical phrase-based model baseline. In our experiments of this paper, we set the beam size $b = 200$ and the beam threshold $\beta = 0$. The maximum initial phrase length is 10.

We use constituency-to-string model (Liu et al., 2006) as the syntax-based model baseline which make use of composed rules (Galley et al., 2006) without handling the unaligned words. In our experiments of this paper, we set the tatTable-limit=20, tatTable-threshold=10^{-1}, stack-limit=100, stack-threshold=10^{-1},hight-limit=3, and length-limit=7.

7.3 Results

We display the results of our experiments in Table 1. Our dependency-to-string model *dep2str* significantly outperforms its constituency structure-based counterpart (*cons2str*) with +1.27 and +1.68 BLEU on MT04 and MT05 respectively. Moreover, without resort to phrases or parse forest, *dep2str* surpasses the hierarchical phrase-based model (*hiero-re*) over +0.53 and +0.4 BLEU on MT04 and MT05 respectively on the basis of a 62% smaller rule set.

Furthermore, We compare some actual translations generated by *cons2str*, *hiero-re* and *dep2str*. Figure 6 shows two translations of our test sets MT04 and MT05, which are selected because each holds a long distance dependency commonly used in Chinese.

In the first example, the Chinese input holds a complex long distance dependencies "巴 尼 耶 在... 与...后 表 示". This dependency corresponds to sentence pattern "noun+prepostional phrase+prepositional phrase+verb", where the former prepositional phrase specifies the position and the latter specifies the time. Both *cons2str* and *hiero-re* are confused by this sentence and mistak-

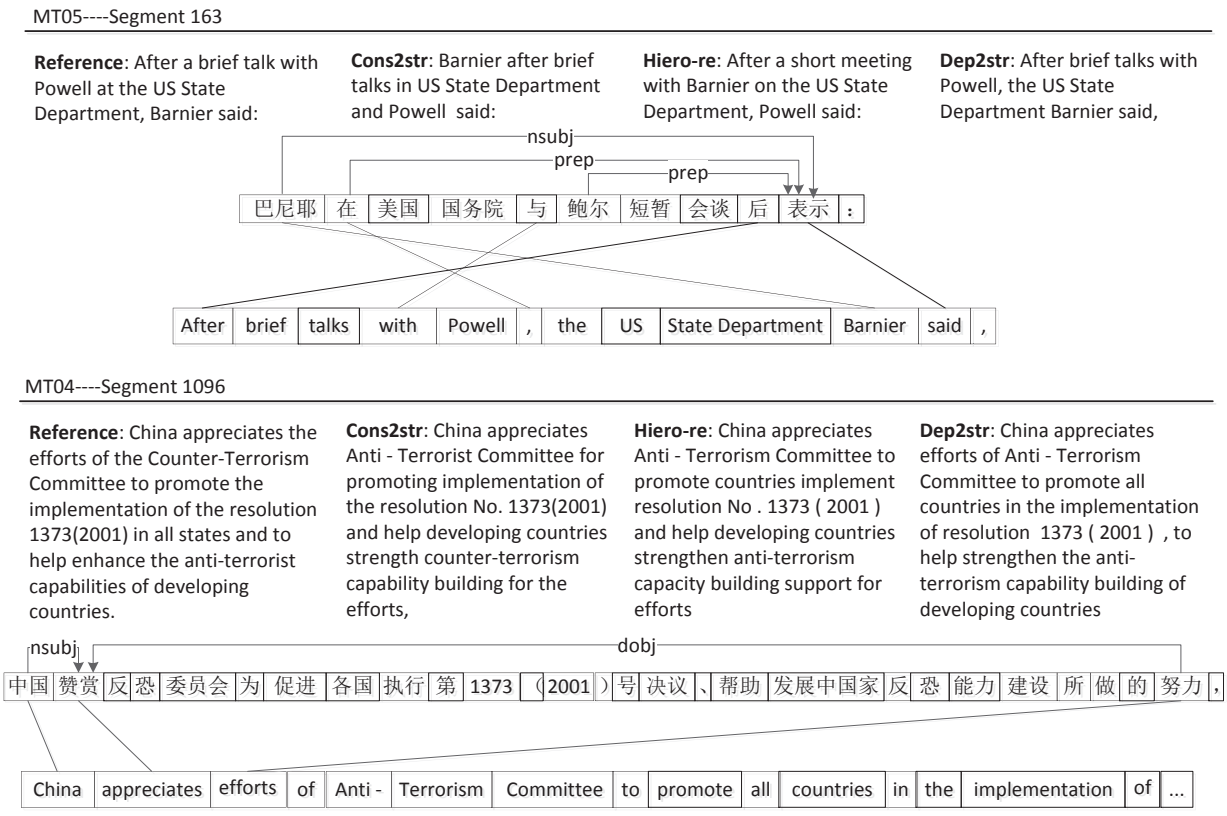


Figure 6: Actual translations produced by the baselines and our system. For our system, we also display the long distance dependencies correspondence in Chinese and English. Here we omit the edges irrelevant to the long distance dependencies.

enly treat "鲍尔(Powell)" as the subjective, thus result in translations with different meaning from the source sentence. Conversely, although "在" is falsely translated into a comma, *dep2str* captures this complex dependency and translates it into "After ... ,(should be *at*) Barnier said", which accords with the reordering of the reference.

In the second example, the Chinese input holds a long distance dependency "中国 赞赏 ... 努力" which corresponds to a simple pattern "noun phrase+verb+noun phrase". However, due to the modifiers of "努力" which contains two sub-sentences including 24 words, the sentence looks rather complicated. *Cons2str* and *hiero-re* fail to capture this long distance dependency and provide monotonic translations which do not reflect the meaning of the source sentence. In contrast, *dep2str* successfully captures this long distance dependency and translates it into "China appreciates efforts of ...", which is almost the same with the reference "China appreciates the efforts of ...".

All these results prove the effectiveness of our dependency-to-string model in both translation and long distance reordering. We believe that the advantage of *dep2str* comes from the characteristics of dependency structures tending to bring semantically related elements together (e.g., verbs become adjacent to all their arguments) and are better suited to lexicalized models (Quirk et al., 2005). And the incapability of *cons2str* and *hiero-re* in handling long distance reordering of these sentences does not lie in the representation of translation rules but the compromises in rule extraction or decoding so as to balance the speed or grammar size and performance. The hierarchical phrase-based model prohibits any nonterminal X from spanning a substring longer than 10 on the source side to make the decoding algorithm asymptotically linear-time (Chiang, 2005).

While constituency structure-based models typically constrain the number of internal nodes (Galley et al., 2006) and/or the height (Liu et al., 2006) of translation rules so as to balance the grammar size and performance. Both strategies limit the ability of the models in processing long distance reordering of sentences with long and complex modification relations.

8 Related Works

As a first step towards semantics, dependency structures are attractive to machine translation. And many efforts have been made to incorporating this desirable knowledge into machine translation.

(Lin, 2004; Quirk et al., 2005; Ding and Palmer, 2005; Xiong et al., 2007) make use of source dependency structures. (Lin, 2004) employs linear paths as phrases and view translation as minimal path covering. (Quirk et al., 2005) extends paths to treelets, arbitrary connected subgraphs of dependency structures, and propose a model based on treelet pairs. Both models require projection of the source dependency structure to the target side via word alignment, and thus can not handle non-isomorphism between languages. To alleviate this problem, (Xiong et al., 2007) presents a dependency treelet string correspondence model which directly map a dependency structure to a target string. (Ding and Palmer, 2005) presents a translation model based on Synchronous Dependency Insertion Grammar(SDIG), which handles some of the non-isomorphism but requires both source and target dependency structures. Most important, all these works do not specify the ordering information directly in translation rules, and resort to either heuristics (Lin, 2004; Xiong et al., 2007) or separate ordering models(Quirk et al., 2005; Ding and Palmer, 2005) to control the word order of translations. By comparison, our model requires only source dependency structure, and handles non-isomorphism and ordering problems simultaneously by directly specifying the ordering information in the head-dependents rules that represent the source side as head-dependents relations and the target side as strings.

(Shen et al., 2008) exploits target dependency structures as dependency language models to ensure the grammaticality of the target string. (Shen et al., 2008) extends the hierarchical phrase-based model and present a string-to-dependency model, which employs string-to-dependency rules whose source side are string and the target as well-formed dependency structures. In contrast, our model exploits source dependency structures, as a tree-based system, it run much faster (linear time vs. cubic time, see (Huang et al., 2006)).

9 Conclusions and future work

In this paper, we present a novel dependency-to-string model, which employs head-dependents rules that represent the source side as head-dependents relations and the target side as string. The head-dependents rules specify the ordering information directly and require only substitution operation. Thus, our model does not need heuristics or ordering model of the previous works to control the word order of translations. Large scale experiments show that our model exhibits good performance in long distance reordering and outperforms the state-of-the-art constituency-to-string model and hierarchical phrase-based model without resort to phrases and parse forest. For the first time, a source dependency-based model shows improvement over the state-of-the-art translation models.

In our future works, we will exploit the semantic information encoded in the dependency structures which is expected to further improve the translations, and replace 1-best dependency structures with dependency forests so as to alleviate the influence caused by parse errors.

Acknowledgments

This work was supported by National Natural Science Foundation of China, Contract 60736014, 60873167, 90920004. We are grateful to the anonymous reviewers for their thorough reviewing and valuable suggestions. We appreciate Yajuan Lv, Wenbin Jiang, Hao Xiong, Yang Liu, Xinyan Xiao, Tian Xia and Yun Huang for the insightful advices in both experiments and writing. Special thanks goes to Qian Chen for supporting my pursuit all through.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of*

ACL 2005, pages 263–270.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL 2005*.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *In Proceedings of EMNLP 2002*, pages 304–311.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of ACL 2006*, pages 961–968, Sydney, Australia, July. Association for Computational Linguistics.

Peter Hellwig. 2006. Parsing with dependency grammars. In *Dependenz und Valenz / Dependency and Valency*, volume 2, pages 1081–1109. Berlin, New York.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL 2007*, pages 144–151, Prague, Czech Republic, June.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. A syntax-directed translator with extended domain of locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 1–8, New York City, New York, June. Association for Computational Linguistics.

Richard Hudson. 1990. *English Word Grammar*. Blackell.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS*, pages 3–10. MIT Press.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Edmonton, Canada, July.

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of Coling 2004*, pages 625–630, Geneva, Switzerland, Aug 23–Aug 27.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of ACL 2006*, pages 609–616, Sydney, Australia, July.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, USA, July.

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

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL-2003*, pages 160–167, Sapporo, Japan, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of ACL 2005*, pages 271–279.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL 2008: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP, volume 30*, pages 901–904.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2007. A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 40–47, Prague, Czech Republic, June.

Arnold M. Zwicky. 1985. Heads. *Journal of Linguistics*, 21:1–29.

Bayesian Checking for Topic Models

David Mimno
Department of Computer Science
Princeton University Princeton, NJ 08540
mimno@cs.princeton.edu

David Blei
Department of Computer Science
Princeton University Princeton, NJ 08540
blei@cs.princeton.edu

Abstract

Real document collections do not fit the independence assumptions asserted by most statistical topic models, but how badly do they violate them? We present a Bayesian method for measuring how well a topic model fits a corpus. Our approach is based on posterior predictive checking, a method for diagnosing Bayesian models in user-defined ways. Our method can identify where a topic model fits the data, where it falls short, and in which directions it might be improved.

1 Introduction

Probabilistic topic models are a suite of machine learning algorithms that decompose a corpus into a set of topics and represent each document with a subset of those topics. The inferred topics often correspond with the underlying themes of the analyzed collection, and the topic modeling algorithm organizes the documents according to those themes.

Most topic models are evaluated by their predictive performance on held out data. The idea is that topic models are fit to maximize the likelihood (or posterior probability) of a collection of documents, and so a good model is one that assigns high likelihood to a held out set (Blei et al., 2003; Wallach et al., 2009).

But this evaluation is not in line with how topic models are frequently used. Topic models seem to capture the underlying themes of a collection—indeed the monicker "topic model" is retrospective—and so we expect that these themes are useful for exploring, summarizing, and learning about its documents (Mimno and McCallum, 2007; Chang et al., 2009). In such exploratory data analysis, however, we are not concerned with the fit to held out data.

In this paper, we develop and study new methods for evaluating topic models. Our methods are based on *posterior predictive checking*, which is a model diagnosis technique from Bayesian statistics (Rubin, 1984; Gelman et al., 1996). The goal of a posterior predictive check (PPC) is to assess the validity of a Bayesian model without requiring a specific alternative model. Given data, we first compute a posterior distribution over the latent variables. Then, we estimate the probability of the observed data under the data-generating distribution that is induced by the posterior (the "posterior predictive distribution"). A data set that is unlikely calls the model into question, and consequently the posterior. PPCs can show where the model fits and doesn't fit the observations. They can help identify the parts of the posterior that are worth exploring.

The key to a posterior predictive check is the *discrepancy function*. This is a function of the data that measures a property of the model which is important to capture. While the model is often chosen for computational reasons, the discrepancy function might capture aspects of the data that are desirable but difficult to model. In this work, we will design a discrepancy function to measure an independence assumption that is implicit in the modeling assumptions but is not enforced in the posterior. We will embed this function in a posterior predictive check and use it to evaluate and visualize topic models in new ways.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 227–237,
Edinburgh, Scotland, UK, July 27–31, 2011.

Specifically, we develop discrepancy functions for latent Dirichlet allocation (the simplest topic model) that measure how well its statistical assumptions about the topics are matched in the observed corpus and inferred topics. LDA assumes that each observed word in a corpus is assigned to a topic, and that the words assigned to the same topic are drawn independently from the same multinomial distribution (Blei et al., 2003). For each topic, we measure the whether this assumption holds by computing the mutual information between the words assigned to that topic and which document each word appeared in. If the assumptions hold, these two variables should be independent: low mutual information indicates that the assumptions hold; high mutual information indicates a mismatch to the modeling assumptions.

We embed this discrepancy in a PPC and study it in several ways. First, we focus on topics that model their observations well; this helps separate interpretable topics from noisy topics (and "boilerplate" topics, which exhibit too little noise). Second, we focus on individual terms within topics; this helps display a model applied to a corpus, and understand which terms are modeled well. Third, we replace the document identity with an external variable that might plausibly be incorporated into the model (such as time stamp or author). This helps point the modeler towards the most promising among more complicated models, or save the effort in fitting one. Finally, we validate this strategy by simulating data from a topic model, and assessing whether the PPC "accepts" the resulting data.

2 Probabilistic Topic Modeling

Probabilistic topic models are statistical models of text that assume that a small number of distributions over words, called "topics," are used to generate the observed documents. One of the simplest topic models is latent Dirichlet allocation (LDA) (Blei et al., 2003). In LDA, a set of K topics describes a corpus; each document exhibits the topics with different proportions. The words are assumed exchangeable within each document; the documents are assumed exchangeable within the corpus.

More formally, let $\phi_1, \ldots, \phi_K$ be K topics, each of which is a distribution over a fixed vocabulary. For each document, LDA assumes the following generative process

1. Choose topic proportions $\theta_d \sim \text{Dirichlet}(\alpha)$.
2. For each word
   (a) Choose topic assignment $z_{d,n} \sim \theta$.
   (b) Choose word $w_{d,n} \sim \phi_{z_{d,n}}$.

This process articulates the statistical assumptions behind LDA: Each document is endowed with its own set of topic proportions θ_d, but the same set of topics $\phi_{1:K}$ governs the whole collection.

Notice that the probability of a word is independent of its document θ_d given its topic assignment $z_{d,n}$ (i.e., $w_{d,n} \perp\!\!\!\perp \theta_d \,|\, z_{d,n}$). Two documents might have different overall probabilities of containing a word from the "vegetables" topic; however, all the words in the collection (regardless of their documents) drawn from that topic will be drawn from the same multinomial distribution.

The central computational problem for LDA is posterior inference. Given a collection of documents, the problem is to compute the conditional distribution of the hidden variables—the topics ϕ_k, topic proportions θ_d, and topic assignments $z_{d,n}$. Researchers have developed many algorithms for approximating this posterior, including sampling methods (Griffiths and Steyvers, 2004) (used in this paper), variational methods (Blei et al., 2003), distributed variants (Asuncion et al., 2008), and online algorithms (Hoffman et al., 2010).

3 Checking Topic Models

Once approximated, the posterior distribution is used for the task at hand. Topic models have been applied to many tasks, such as classification, prediction, collaborative filtering, and others. We focus on using them as an exploratory tool, where we assume that the topic model posterior provides a good decomposition of the corpus and that the topics provide good summaries of the corpus contents.

But what is meant by "good"? To answer this question, we turn to Bayesian model checking (Rubin, 1981; Gelman et al., 1996). The goal of Bayesian model checking is to assess whether the observed data matches the modeling assumptions in the directions that are important to the analysis. The

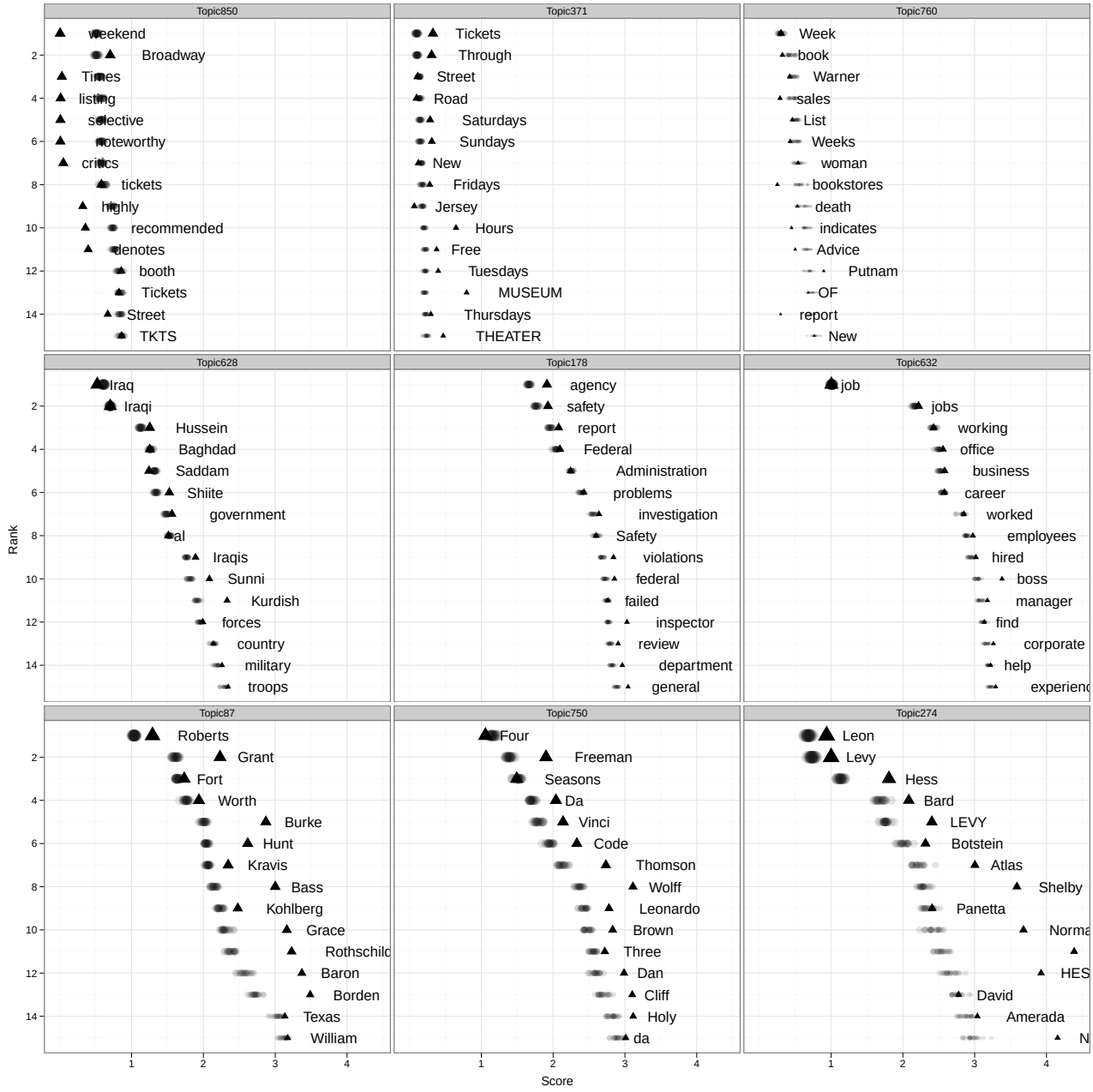


Figure 1: **Visualization of variability within topics.** Nine randomly selected topics from the New York Times with low (top row), medium (middle row) and high (bottom row) mutual information between words and documents. The y-axis shows term rank within the topic, with size proportional to log probability. The x-axis represents divergence from the multinomial assumption for each word: terms that are uniformly distributed across documents are towards the left, while more specialized terms are to the right. Triangles represent real values, circles represent 20 replications of this same plot from the posterior of the model.

intuition is that only when satisfied with the model should the modeler use the posterior to learn about her data. In complicated Bayesian models, such as topic models, Bayesian model checking can point to the parts of the posterior that better fit the observed data set and are more likely to suggest something meaningful about it.

In particular, we will develop posterior predictive checks (PPC) for topic models. In a PPC, we specify a *discrepancy function*, which is a function of the data that measures an important property that we want the model to capture. We then assess whether the observed value of the function is similar to values of the function drawn from the posterior, through the distribution of the data that it induces. (This distribution of the data is called the "posterior predictive distribution.")

An innovation in PPCs is the *realized discrepancy function* (Gelman et al., 1996), which is a function of the data and any hidden variables that are in the model. Realized discrepancies induce a traditional discrepancy by marginalizing out the hidden variables. But they can also be used to evaluate assumptions about *latent* variables in the posterior, especially when combined with techniques like MCMC sampling that provide realizations of them. In topic models, as we will see below, we use a realized discrepancy to factor the observations and to check specific components of the model that are discovered by the posterior.

3.1 A realized discrepancy for LDA

Returning to LDA, we design a discrepancy function that checks the independence assumption of words given their topic assignments. As we mentioned above, given the topic assignment z the word w should be independent of its document θ. Consider a decomposition of a corpus from LDA, which assigns every observed word $w_{d,n}$ to a topic $z_{d,n}$. Now restrict attention to all the words assigned to the kth topic and form two random variables: W are the words assigned to the topic and D are the document indices of the words assigned to that topic. If the LDA assumptions hold then knowing W gives no information about D because the words are drawn independently from the topic.

We measure this independence with the mutual information between W and D:[1]

$$\begin{aligned} &MI(W, D \mid k) \\ &= \sum_w \sum_d P(w, d \mid k) \log \frac{P(w \mid d, k) P(d \mid k)}{P(w \mid k) P(d \mid k)} \\ &= \sum_w \sum_d \frac{N(w, d, k)}{N(k)} \log \frac{N(w, d, k) N(k)}{N(d, k) N(w, k)}. \end{aligned} \quad (1)$$

Where $N(w, d, k)$ is the number of tokens of type w in topic k in document d, with $N(w, k) = \sum_d N(w, d, k)$, $N(d, k) = \sum_w N(w, d, k)$, and $N(k) = \sum_{w,d} N(w, d, k)$. This function measures the divergence between the joint distribution over word and document index and the product of the marginal distributions. In the limit of infinite samples, independent random variables have mutual information of zero, but we expect finite samples to have non-zero values even for truly independent variables. Notice that this is a realized discrepancy; it depends on the latent assignments of observed words to topics.

Eq. 1 is defined as a sum over a set of documents and a set of words. We can rearrange this summation as a weighted sum of the *instantaneous mutual information* between words and documents:

$$IMI(w, D \mid k) = H(D|k) - H(D \mid W = w, k). \quad (2)$$

This quantity can be understood by considering the per-topic distribution of document labels, $p(d|k)$. This distribution is formed by normalizing the counts of how many words assigned to topic k appeared in each document. The first term of Eq. 2 is the entropy—some topics are evenly distributed across many documents (high entropy); others are concentrated in fewer documents (low entropy).

The second term conditions this distribution on a particular word type w by normalizing the per-document number of times w appeared in each document (in topic k). If this distribution is close to $p(d|k)$ then $H(D|W = w, k)$ will be close to $H(D|k)$ and $IMI(w, D|k)$ will be low. If, on the other hand, word w occurs many times in only a few documents, it will have lower entropy over docu-

[1] There are other choices of discrepancies, such as word-word point-wise mutual information scores (Newman et al., 2010).

ments than the overall distribution over documents for the topic and $IMI(w, D|k)$ will be high.

We illustrate this discrepancy in Figure 1, which shows nine topics trained from the *New York Times*.[2] Each row contains randomly selected topics from low, middle, and high ranges of MI, respectively. Each triangle represents a word. Its place on the y-axis is its rank in the topic. Its place on the x-axis is its $IMI(w|k)$, with more uniformly distributed words (low IMI) to the left and more specific words (high IMI) to the right. (For now, ignore the other points in this figure.) IMI varies between topics, but tends to increase with rank as less frequent words appear in fewer documents.

The discrepancy captures different kinds of structure in the topics. The top left topic represents formulaic language, language that occurs verbatim in many documents. In particular, it models the boilerplate text "Here is a selective listing by critics of The Times of new or noteworthy..." Identifying repeated phrases is a common phenomenon in topic models. Most words show lower than expected IMI, indicating that word use in this topic is less variable than data drawn from a multinomial distribution. The middle-left topic is an example of a good topic, according to this discrepancy, which is related to Iraqi politics. The bottom-left topic is an example of the opposite extreme from the top-left. It shows a loosely connected series of proper names with no overall theme.

3.2 Posterior Predictive Checks for LDA

Intuitively, the middle row of topics in Figure 1 are the sort of topics we look for in a model, while the top and bottom rows contain topics that are less useful. Using a PPC, we can formally measure the difference between these topics. For each of the real topics in Figure 1 we regenerated the same figure 20 times. We sampled new words for every token from the posterior distribution of the topic, and recalculated the rank and IMI for each word. These "shadow" figures are shown as gray circles. The density of those circles creates a reference distribution indicating the expected IMI values at each rank under the multinomial assumption.

[2]Details about the corpus and model fitting are in Section 4.2. Similar figures for two other corpora are in the supplement.

By themselves, IMI scores give an indication of the distribution of a word between documents within a topic: small numbers are better, large numbers indicate greater discrepancy. These scores, however, are based on the specific allocation of words to topics. For example, lower-ranked, less frequent words within a topic tend to have higher IMI scores than higher-ranked, more frequent words. This difference may be due to greater violation of multinomial assumptions, but may also simply be due to smaller sample sizes, as the entropy $H(D|W = w, k)$ is estimated from fewer tokens. The reference distributions help distinguish between these two cases.

In more detail, we generate replications of the data by considering a Gibbs sampling state. This state assigns each observed word to a topic. We first record the number of instances of each term assigned to each topic, $N(w|k)$. Then for each word $w_{d,n}$ in the corpus, we sample a new observed word $w_{d,n}^{rep}$ where $P(w) \propto N(w|z_{d,n})$. (We did not use smoothing parameters.) Finally, we recalculate the mutual information and instantenous mutual information for each topic.

In the top-left topic, most of the words have much lower IMI than the word at the same rank in replications, indicating lower than expected variability. The exception is the word *Broadway*, which is more variable than expected. In the middle-left topic, IMI for the words *Iraqi* and *Baghdad* occur within the expected range. These words fit the multinomial assumption: any word assigned to this topic is equally likely to be *Iraqi*. Values for the words *Shiite*, *Sunni*, and *Kurdish* are more specific to particular documents than we expect under the model. In the bottom-left topic, almost all words occur with greater variability than expected. This topic combines many terms with only coincidental similarity, such as Mets pitcher Grant Roberts and the firm Kohlberg Kravis Roberts.

Turning to an analysis of the full mutual information, Figure 2 shows the three left-hand topics from Figure 1: *Weekend*, *Iraq*, and *Roberts*. The histogram represents MI scores for 100 replications of the topic, rescaled to have mean zero and unit variance. The observed value, also rescaled, and the mean replicated value (set to zero) are shown with vertical lines. The formulaic *Weekend* topic has significantly lower than expected MI. The *Iraq*

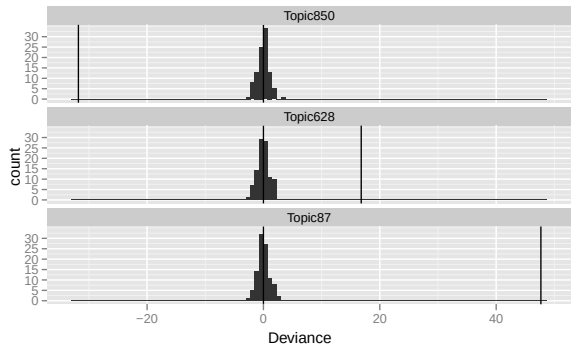


Figure 2: **News: Observed topic scores (vertical lines) relative to replicated scores, rescaled so that replications have zero mean and unit variance.** The *Weekend* topic (top) has lower than expected MI. The *Iraq* (middle) and *Roberts* (bottom) topics both have MI greater than expected.

and *Roberts* topics have significantly greater than expected MI.

For most topics the actual discrepancy is outside the range of any replicated discrepancies. In their original formulation, PPCs prescribe computing a tail probability of a replicated discrepancy being greater than (or less than) the observed discrepancy under the posterior predictive distribution. For example if an observed value is greater than 70 of 100 replicated values, we report a PPC p-value of 0.7.

When the observed value is far outside the range of any replicated values, as in Figure 2, that tail probability will be degenerate at 0 or 1. So, we report instead a *deviance* value, an alternative way of comparing an observed value to a reference distribution. We compute the distribution of the replicated discrepancies and compute its standard deviation. We then compute how many standard deviations the observed discrepancy is from the mean of the replicated discrepancies.

This score allows us to compare topics. The observed value for the *Weekend* topic is 31.8 standard deviations below the mean replicated value, and thus has deviance of -31.8, which is lower than expected. The *Iraq* topic has deviance of 16.8 and the *Roberts* topic has deviance of 47.7. This matches our intuition that the former topic is more useful than the latter.

4 Searching for Systematic Deviations

We demonstrated that the mutual information discrepancy function can detect violations of multinomial assumptions, in which instances of a term in a given topic are not independently distributed among documents. One way to address this lack of fit is to encode document-level extra-multinomial variance ("burstiness") into the model using Dirichlet compound multinomial distributions (Doyle and Elkan, 2009). If there is no pattern to the deviations from multinomial word use across documents, this method is the best we can do.

In many corpora, however, there are systematic deviations that can be explained by additional variables. LDA is the simplest generative topic model, and researchers have developed many variants of LDA that account for a variety of variables that can be found or measured with a corpus. Examples include models that account for time (Blei and Lafferty, 2006), books (Mimno and McCallum, 2007), and aspect or perspective (Mei and Zhai, 2006; Lin et al., 2008; Paul et al., 2010). In this section, we show how we can use the mutual information discrepancy function of Equation 1 and PPCs to guide our choice in which topic model to fit.

Greater deviance implies that a particular grouping better explains the variation in word use within a topic. The discrepancy functions are large when words appear more than expected in some groups and less than expected in others. We know that the individual documents show significantly more variation than we expect from replications from the model's posterior distribution. If we combine documents randomly in a meaningless grouping, such deviance should decrease, as differences between documents are "smoothed out." If a grouping of documents shows equal or greater deviation, we can assume that that grouping is maintaining the underlying structure of the systematic deviation from the multinomial assumption, and that further modeling or visualization using that grouping might be useful.

4.1 PPCs for systematic discrepancy

The idea is that the words assigned to a topic should be independent of both document and any other variable that might be associated with the document. We simply replace the document index d with another

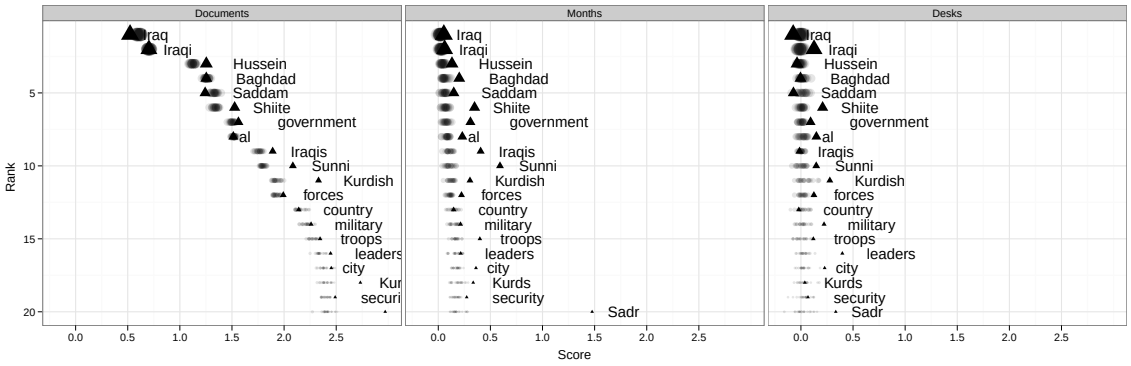


Figure 3: **Groupings decrease MI, but values are still larger than expected.** Three ways of grouping words in a topic from the *New York Times*. The word *leaders* varies more between desks than by time, while *Sadr* varies more by time than desk.

variable g in the discrepancy. For example, the New York Times articles are each associated with a particular news desk and also associated with a time stamp. If the topic modeling assumptions hold, the words are independent of both these variables. If we see a significant discrepancy relative to a grouping defined by a metadata feature, this systematic variability suggests that we might want to take that feature into account in the model.

Let $\mathcal{G}$ be a set of groups and let $\gamma \in \mathcal{G}^D$ be a grouping of D documents. Let $N(w, g, k) = \sum_d N(w, d, k) I_{\gamma_d = g}$, that is, the number of words of type w in topic k in documents in group g, and define the other count variables similarly. We can now substitute these group-specific counts for the document-specific counts in the discrepancy function in Eq. 1. Note that the previous discrepancy functions are equivalent to a trivial grouping, in which each document is the only member of its own group. In the following experiments we explore groupings by published volume, blog, preferred political candidate, and newspaper desk, and evaluate the effect of those groupings on the deviation between mean replicated values and observed values of those functions.

4.2 Case studies

We analyze three corpora, each with its own metadata: the New York Times Annotated Corpus (1987–2007)[3], the CMU 2008 political blog corpus (Eisenstein and Xing, 2010), and speeches from the British House of Commons from 1830–1891.[4] Descriptive statistics are presented in Table 1. The realization is represented by a single Gibbs sampling state after 1000 iterations of Gibbs sampling.

Table 1: Statistics for models used as examples.

Name	Docs	Tokens	Vocab	Topics
News	1.8M	76M	121k	1000
Blogs	13k	2.2M	90k	100
Parliament	540k	55M	52k	300

New York Times articles. Figure 3 shows three groupings of words for the middle-left topic in Figure 1: by document, by month of publication (e.g. May of 2005), and by desk (e.g. Editorial, Foreign, Financial). Instantaneous mutual information values are significantly smaller for the larger groupings, but the actual values are still larger than expected under the model. We are interested in measuring the degree to which word usage varies within topics as a function of both time and the perspective of the article. For example, we may expect that word choice may differ between opinion articles, which overtly reflect an author's views, and news articles, which take a more objective, factual approach.

We summarize each grouping by plotting the distribution of deviance scores for all topics. Results for all 1000 topics grouped by documents, months, and desks are shown in Figure 4.

[3] http://www.ldc.upenn.edu

[4] http://www.hansard-archive.parliament.uk/

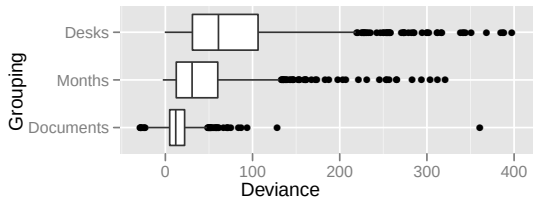


Figure 4: **News: Lack of fit correlates best with desks.** We calculate the number of standard deviations between the mean replicated discrepancy and the actual discrepancy for each topic under three groupings. Boxes represent typical ranges, points represent outliers.

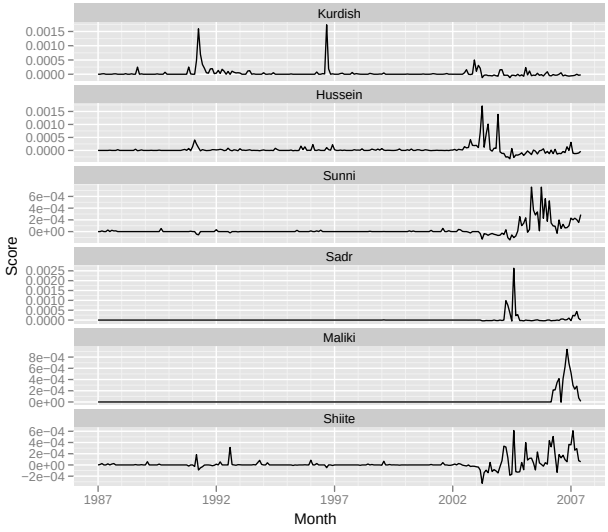


Figure 5: **News: Events change word distributions.** Words with the largest MI from a topic on Iraq's government are shown, with individual scores grouped by month.

Finally, we can analyze how individual words interact with groupings like time or desk. Figure 5 breaks down the per-word discrepancy shown in Figure 3 by month, for the words with the largest overall discrepancy. *Kurdish* is prominent during the Gulf War and the 1996 cruise missile strikes, but is less significant during the Iraq War. Individuals (*Hussein*, *Sadr*, and *Maliki*) move on and off the stage.

Political blogs. The CMU 2008 political blog corpus consists of six blogs, three of which supported Barack Obama and three of which supported John McCain. This corpus has previously been considered in the context of aspect-based topic models (Ahmed and Xing, 2010) that assign distinct word distributions to liberal and conservative bloggers. It is reasonable to expect that blogs with different political leanings will use measurably different language to describe the same themes, suggesting that there will be systematic deviations from a multinomial hypothesis of exchangeability of words within topics. Indeed, Ahmed and Xing obtained improved results with such a model. Figure 6 shows the distribution of standard deviations from the mean replicated value for a set of 150 topics grouped by document, blog, and preferred candidate. Deviance is greatest for blogs, followed by candidates and then documents.

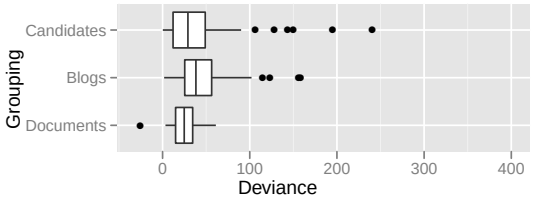


Figure 6: **Blogs: Lack of fit correlates more with blog than preferred candidate.** Grouping by preferred candidate has only slightly higher average deviance than by documents, but the variance is greater.

Grouping by blogs appears to show greater deviance from mean replicated values than grouping by candidates, indicating that there is further structure in word choice beyond a simple liberal/conservative split. Are these results, however, comparable? It may be that this difference is explained by the fact that there are six blogs and only

two candidates. To determine whether this particular assignment of documents to blogs is responsible for the difference in discrepancy functions or whether any such split would have greater deviance, we compared random groupings to the real groupings and recalculate the PPC. We generated 10 such groupings by permuting document blog labels and another 10 by permuting document candidate labels, each time holding the topics fixed. The average number of standard deviations across topics was 6.6 ± 14.4 for permuted "candidates" compared to 37.9 ± 39.2 for the real corpus, and 10.6 ± 12.9 for permuted "blogs" compared to 44.4 ± 29.6 for real blogs.

British parliament proceedings. The parliament corpus is divided into 305 volumes, each comprising about three weeks of debates, with between 600 and 4000 speeches per session. In addition to volumes, 10 Prime Ministers were in office during this period.

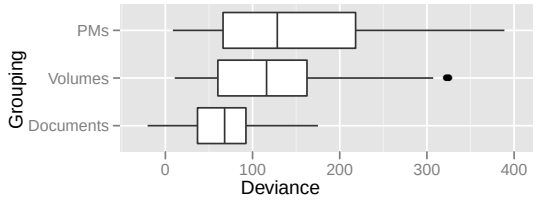


Figure 7: **Parliament: Lack-of-fit correlates with time (publication volume).** Correlation with prime ministers is not significantly better than with volume.

Grouping by prime minister shows greater average deviance than grouping by volumes, even though there are substantially fewer divisions. Although such results would need to be accompanied by permutation experiments as in the blog corpus, this methodology may be of interest to historians.

In order to provide insight into the nature of temporal variation, we can group the terms in the summation in Equation 1 by word and rank the words by their contribution to the discrepancy function. Figure 8 shows the most "mismatching" words for a topic with the most probable words *ships, vessels, admiralty, iron, ship, navy*, consistent with changes in naval technology during the Victorian era (that is, wooden ships to "iron clads"). Words that occur more prominently in the topic (*ships, vessels*) are also variable, but more consistent across time.

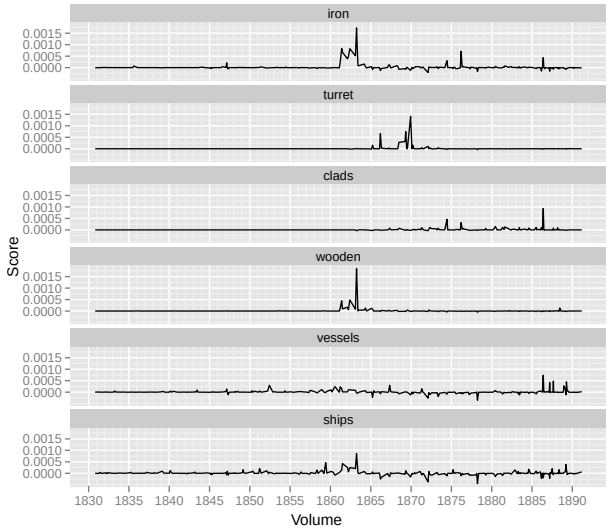


Figure 8: **Parliament: *iron-clads* introduced in 1860s.** High probability words (*ships, vessels*) are variable, but show less concentrated discrepancy than *iron, wooden*.

5 Calibration on Synthetic Data

A posterior predictive check asks "do observations sampled from the learned model look like the original data?" In the previous sections, we have considered PPCs that explore variability within a topic on a per-word basis, measure discrepancy at the topic level, and compare deviance over all topics between groupings of documents. Those results show that the PPC detects deviation from multinomial assumptions when it exists: as expected, variability in word choice aligns with known divisions in corpora, for example by time and author perspective. We now consider the opposite direction. When documents are generated from a multinomial topic model, PPCs should not detect systematic deviation.

We must also distinguish between lack of fit due to model misspecification and lack of fit due to approximate inference. In this section, we present synthetic data experiments where the learned model is precisely the model used to generate documents. We show that there is significant lack of fit introduced by approximate inference, which can be corrected by considering only parts of the model that are well-estimated.

We generated 10 synthetic corpora, each consisting of 100,000 100-word documents, drawn from 20

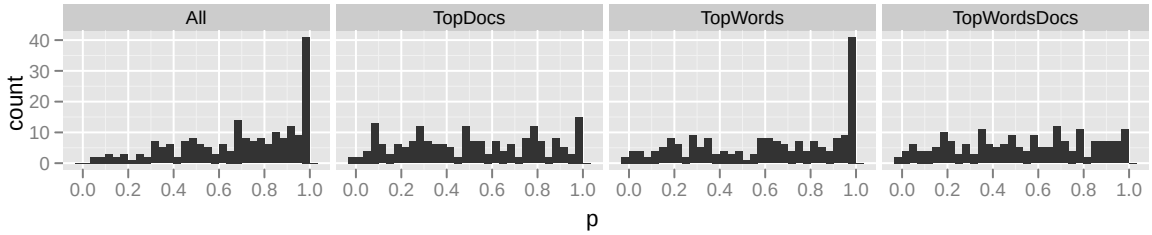


Figure 9: **Replicating only documents with large allocation in the topic leads to more uniform p-values.** p-values for 200 topics estimated from synthetic data generated from an LDA model are either uniform or skewed towards 1.0. Overly conservative p-values would be clustered around 0.5.

topics over a vocabulary of 100 terms. Hyperparameters for both the document-topic and topic-term Dirichlet priors were 0.1 for each dimension. We then trained a topic model with the same hyperparameters and number of topics on each corpus, saving a Gibbs sampling state.

We can measure the fit of a PPC by examining the distribution of empirical p-values, that is, the proportion of replications $\boldsymbol{w}^{rep}$ that result in discrepancies less than the observed value. p-values should be uniformly distributed on $(0, 1)$. Non-uniform p-values indicate a lack of *calibration*. Unlike real collections, in synthetic corpora the range of discrepancies from these replicated collections often includes the real values, so p-values are meaningful. A histogram of p-values for 200 synthetic topics after 100 replications is shown in the left panel of Figure 9.

PPCs have been criticized for reusing training data for model checking. For some models, the posterior distribution is too close to the data, so all replicated values are close to the real value, leading to p-values clustered around 0.5 (Draper and Krnjajic, 2006; Bayarri and Castellanos, 2007). We test divergence from a uniform distribution with a Kolmogorov-Smirnov test. Our results indicate that LDA is not overfitting, but that the distribution is not uniform (KS $p < 0.00001$).

The PPC framework allows us to choose discrepancy functions that reflect the relative importance of subsets of words and documents. The second panel in Figure 9 sums only over the 20 documents with the largest probability of the topic, the third sums over all documents but only over the top 10 most probable words, and the fourth sums over only the top words and documents. This test indicates that the distribution of p-values for the subset *TopWords* is not uniform (KS $p < 0.00001$), but that a uniform distribution is a good fit for *TopDocs* (KS $p = 0.358$) and *TopWordsDocs* (KS $p = 0.069$).

6 Conclusions

We have developed a Bayesian model checking method for probabilistic topic models. Conditioned on their topic assignment, the words of the documents are independently and identically distributed by a multinomial distribution. We developed a realized discrepancy function—the mutual information between words and document indices, conditioned on a topic—that checks this assumption. We embedded this function in a posterior predictive check.

We demonstrated that we can use this posterior predictive check to identify particular topics that fit the data, and particular topics that misfit the data in different ways. Moreover, our method provides a new way to visualize topic models.

We adapted the method to corpora with external variables. In this setting, the PPC provides a way to guide the modeler in searching through more complicated models that involve more variables.

Finally, on simulated data, we demonstrated that PPCs with the mutual information discrepancy function can identify model fit and model misfit.

Acknowledgments

David M. Blei is supported by ONR 175-6343, NSF CAREER 0745520, AFOSR 09NL202, the Alfred P. Sloan foundation, and a grant from Google. David Mimno is supported by a Digital Humanities Research grant from Google. Arthur Spirling and Andy

Eggers suggested the use of the Hansards corpus.

References

Amr Ahmed and Eric Xing. 2010. Staying informed: Supervised and semi-supervised multi-view topical analysis of ideological perspective. In *EMNLP*.

Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Asynchronous distributed learning of topic models. In *NIPS*.

M.J. Bayarri and M.E. Castellanos. 2007. Bayesian checking of the second levels of hierarchical models. *Statistical Science*, 22(3):322–343.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *ICML*.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 22*, pages 288–296.

Gabriel Doyle and Charles Elkan. 2009. Accounting for burstiness in topic models. In *ICML*.

David Draper and Milovan Krnjajic. 2006. Bayesian model specification. Technical report, University of California, Santa Cruz.

Jacob Eisenstein and Eric Xing. 2010. The CMU 2008 political blog corpus. Technical report, Carnegie Mellon University.

A. Gelman, X.L. Meng, and H.S. Stern. 1996. posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica*, 6:733–807.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *PNAS*, 101(suppl. 1):5228–5235.

Matthew Hoffman, David Blei, and Francis Bach. 2010. Online learning for latent dirichlet allocation. In *NIPS*.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *PKDD*.

Qiaozhu Mei and ChengXiang Zhai. 2006. A mixture model for contextual text mining. In *KDD*.

David Mimno and Andrew McCallum. 2007. Organizing the OCA: learning faceted subjects from a library of digital books. In *JCDL*.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Michael J. Paul, ChengXiang Zhai, and Roxana Girju. 2010. Summarizing contrastive viewpoints in opinionated text. In *EMNLP*.

Donald B. Rubin. 1981. Estimation in parallel randomized experiments. *Journal of Educational Statistics*, 6:377–401.

D. Rubin. 1984. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, 12(4):1151–1172.

Hanna Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *ICML*.

Dual Decomposition with Many Overlapping Components

André F. T. Martins[*†] **Noah A. Smith**[*] **Pedro M. Q. Aguiar**[‡] **Mário A. T. Figueiredo**[†]
[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[‡]Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal
[†]Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal
{afm,nasmith}@cs.cmu.edu, aguiar@isr.ist.utl.pt, mtf@lx.it.pt

Abstract

Dual decomposition has been recently proposed as a way of combining complementary models, with a boost in predictive power. However, in cases where lightweight decompositions are not readily available (*e.g.*, due to the presence of rich features or logical constraints), the original subgradient algorithm is inefficient. We sidestep that difficulty by adopting an augmented Lagrangian method that accelerates model consensus by regularizing towards the averaged votes. We show how first-order logical constraints can be handled efficiently, even though the corresponding subproblems are no longer combinatorial, and report experiments in dependency parsing, with state-of-the-art results.

1 Introduction

The last years have witnessed increasingly accurate models for syntax, semantics, and machine translation (Chiang, 2007; Finkel et al., 2008; Petrov and Klein, 2008; Smith and Eisner, 2008; Martins et al., 2009a; Johansson and Nugues, 2008; Koo et al., 2010). The predictive power of such models stems from their ability to break locality assumptions. The resulting combinatorial explosion typically demands some form of approximate decoding, such as sampling, heuristic search, or variational inference.

In this paper, we focus on parsers built from linear programming relaxations, the so-called "turbo parsers" (Martins et al., 2009a; Martins et al., 2010). Rush et al. (2010) applied dual decomposition as a way of combining models which alone permit efficient decoding, but whose combination is intractable. This results in a relaxation of the original problem that is elegantly solved with the subgradient algorithm. While this technique has proven quite effective in parsing (Koo et al., 2010; Auli and Lopez, 2011) as well as machine translation (Rush and Collins, 2011), we show here that its success is strongly tied to the ability of finding a "good" decomposition, *i.e.*, one involving few overlapping components (or *slaves*). With many components, the subgradient algorithm exhibits extremely slow convergence (cf. Fig. 2). Unfortunately, a lightweight decomposition is not always at hand, either because the problem does not factor in a natural way, or because one would like to incorporate features that cannot be easily absorbed in few tractable components. Examples include features generated by statements in first-order logic, features that violate Markov assumptions, or history features such as the ones employed in transition-based parsers.

To tackle the kind of problems above, we adopt DD-ADMM (Alg. 1), a recently proposed algorithm that accelerates dual decomposition (Martins et al., 2011). DD-ADMM retains the modularity of the subgradient-based method, but it speeds up consensus by regularizing each slave subproblem towards the averaged votes obtained in the previous round (cf. Eq. 14). While this yields more involved subproblems (with a quadratic term), we show that exact solutions can still be efficiently computed for all cases of interest, by using sort operations. As a result, we obtain parsers that can handle very rich features, do not require specifying a decomposition, and can be heavily parallelized. We demonstrate the success of the approach by presenting experiments in dependency parsing with state-of-the-art results.

2 Background

2.1 Structured Prediction

Let $x \in \mathcal{X}$ be an input object (*e.g.*, a sentence), from which we want to predict a structured output $y \in \mathcal{Y}$ (*e.g.*, a parse tree). The output set $\mathcal{Y}$ is assumed too large for exhaustive search to be tractable. We assume to have a model that assigns a score $f(y)$ to each candidate output, based on which we predict

$$\widehat{y} = \arg\max_{y \in \mathcal{Y}} f(y). \quad (1)$$

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 238–249,
Edinburgh, Scotland, UK, July 27–31, 2011.

Designing the model must obey certain practical considerations. If efficiency is the major concern, a simple model is usually chosen so that Eq. 1 can be solved efficiently, at the cost of limited expressive power. If we care more about accuracy, a model with richer features and more involved score functions may be designed. Decoding, however, will be more expensive, and approximations are often necessary. A typical source of intractability comes from the combinatorial explosion inherent in the composition of two or more tractable models (Bar-Hillel et al., 1964; Tromble and Eisner, 2006). Recently, Rush et al. (2010) have proposed a dual decomposition framework to address NLP problems in which the global score decomposes as $f(y) = f_1(z_1) + f_2(z_2)$, where z_1 and z_2 are two overlapping "views" of the output, so that Eq. 1 becomes:

$$\begin{array}{ll} \text{maximize} & f_1(z_1) + f_2(z_2) \\ \text{w.r.t.} & z_1 \in \mathcal{Y}_1, z_2 \in \mathcal{Y}_2 \\ \text{s.t.} & z_1 \sim z_2. \end{array} \quad (2)$$

Above, the notation $z_1 \sim z_2$ means that z_1 and z_2 "agree on their overlaps," and an isomorphism $\mathcal{Y} \simeq \{\langle z_1, z_2 \rangle \in \mathcal{Y}_1 \times \mathcal{Y}_2 \mid z_1 \sim z_2\}$ is assumed. We next formalize these notions and proceed to compositions of an *arbitrary* number of models. Of special interest is the unexplored setting where this number is very large and each component very simple.

2.2 Decomposition into Parts

A crucial step in the design of structured predictors is that of decomposing outputs into parts (Taskar et al., 2003). We assume the following setup:

Basic parts. We let $\mathcal{R}$ be a set of *basic parts*, such that each element $y \in \mathcal{Y}$ can be identified with a subset of $\mathcal{R}$. The exact meaning of a "basic part" is problem dependent. For example, in dependency parsing, $\mathcal{R}$ can be the set of all possible dependency arcs (see Fig. 1); in phrase-based parsing, it can be the set of possible spans; in sequence labeling, it can be the set of possible labels at each position. Our only assumption is that we can "read out" y from the basic parts it contains. For convenience, we represent y as a binary vector, $y = \langle y(r) \rangle_{r \in \mathcal{R}}$, where $y(r) = 1$ if part r belongs to y, and 0 otherwise.

Decomposition. We generalize the decomposition in Eq. 2 by considering sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_S$ for $S \geq 2$.

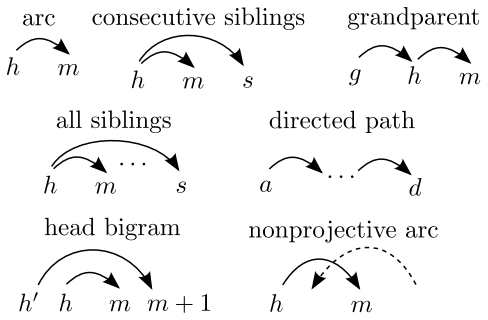


Figure 1: Parts used by our parser. *Arcs* are the basic parts: any dependency tree can be "read out" from the arcs it contains. *Consecutive siblings* and *grandparent* parts introduce horizontal and vertical Markovization (McDonald et al., 2006; Carreras, 2007). We break the horizontal Markov assumption via *all siblings* parts and the vertical one through parts which indicate a *directed path* between two words. Inspired by transition-based parsers, we also adopt *head bigram* parts, which look at the heads attached to consecutive words. Finally, we follow Martins et al. (2009a) and have parts which indicate if an arc is *non-projective* (*i.e.*, if it spans words that do not descend from its head).

Each $\mathcal{Y}_s$ is associated with its own set of parts $\mathcal{R}_s$, in the same sense as above; we represent the elements of $\mathcal{Y}_s$ as binary vectors $z_s = \langle z_s(r) \rangle_{r \in \mathcal{R}_s}$. Examples are vectors indicating a tree structure, a sequence, or an assignment of variables to a factor, in which case it may happen that only some binary vectors are legal. Some parts in $\mathcal{R}_s$ are basic, while others are not. We denote by $\bar{\mathcal{R}}_s = \mathcal{R}_s \cap \mathcal{R}$ the subset of the ones that are. In addition, we assume that:

- $\mathcal{R}_1, \ldots, \mathcal{R}_S$ jointly cover $\mathcal{R}$, *i.e.*, $\mathcal{R} \subseteq \bigcup_{s=1}^{S} \mathcal{R}_s$;
- Only basic parts may overlap, *i.e.*, $\mathcal{R}_s \cap \mathcal{R}_t \subseteq \mathcal{R}, \forall s, t \in \{1, \ldots, S\}$;
- Each $z_s \in \mathcal{Y}_s$ is completely defined by its entries indexed by elements of $\bar{\mathcal{R}}_s$, from which we can guess the ones in $\mathcal{R}_s \setminus \bar{\mathcal{R}}_s$. This implies that each $y \in \mathcal{Y}$ has a unique decomposition $\langle z_1, \ldots, z_S \rangle$.

Fig. 1 shows several parts used in dependency parsing models; in phrase-based parsing, these could be spans and production rules anchored in the surface string; in sequence labeling, they can be unigram, bigram, and trigram labels.[1]

[1]There is a lot of flexibility about how to decompose the model into S components: each set $\mathcal{R}_s$ can correspond to a sin-

Global consistency. We want to be able to read out $y \in \mathcal{Y}$ by "gluing" together the components $\langle z_1, \ldots, z_S \rangle$. This is only meaningful if they are "globally consistent," a notion which we make precise. Two components $z_s \in \mathcal{Y}_s$ and $z_t \in \mathcal{Y}_t$ are said to be *consistent* (denoted $z_s \sim z_t$) if they agree on their overlaps, *i.e.*, if $z_s(r) = z_t(r), \forall r \in \mathcal{R}_s \cap \mathcal{R}_t$. A complete assignment $\langle z_1, \ldots, z_S \rangle$ is *globally consistent* if all pairs of components are consistent. This is equivalent to the existence of a witness vector $\langle u(r) \rangle_{r \in \mathcal{R}}$ such that $z_s(r) = u(r), \forall s, r \in \bar{\mathcal{R}}_s$.

With this setup, assuming that the score function decomposes as $f(z) = \sum_{s=1}^{S} f_s(z_s)$, the decoding problem (which extends Eq. 2 for $S \geq 2$) becomes:

$$\begin{array}{lll} P: & \text{maximize} & \sum_{s=1}^{S} f_s(z_s) \\ & \text{w.r.t.} & z_s \in \mathcal{Y}_s, \quad \forall s \\ & & \langle u(r) \rangle_{r \in \mathcal{R}} \in \mathbb{R}^{|\mathcal{R}|}, \\ & \text{s.t.} & z_s(r) = u(r), \quad \forall s, r \in \bar{\mathcal{R}}_s. \end{array} \tag{3}$$

We call the equality constraints expressed in the last line the "agreement constraints." It is these constraints that complicate the problem, which would otherwise be exactly separable into S subproblems. The dual decomposition method (Komodakis et al., 2007; Rush et al., 2010) builds an approximation by dualizing out these constraints, as we describe next.

2.3 Dual Decomposition

We describe dual decomposition in a slightly different manner than Rush et al. (2010): we will first build a relaxation of P (called P'), in which the entire approximation is enclosed. Then, we dualize P', yielding problem D. In the second step, the duality gap is zero, *i.e.*, P' and D are equivalent.[2]

Relaxation. For each $s \in \{1, \ldots, S\}$ we consider the convex hull of $\mathcal{Y}_s$,

$$\mathcal{Z}_s = \left\{ \sum_{z_s \in \mathcal{Y}_s} p(z_s) z_s \;\middle|\; p(z_s) \geq 0, \sum_{z_s \in \mathcal{Y}_s} p(z_s) = 1 \right\}. \tag{4}$$

We have that $\mathcal{Y}_s = \mathcal{Z}_s \cap \mathbb{Z}^{|\mathcal{R}_s|}$; hence, problem P (Eq. 3) is equivalent to one in which each $\mathcal{Y}_s$ is replaced by $\mathcal{Z}_s$ and the z-variables are constrained to be integer. By dropping the integer constraints, we obtain the following relaxed problem:

$$\begin{array}{lll} P': & \text{maximize} & \sum_{s=1}^{S} f_s(z_s) \\ & \text{w.r.t.} & z_s \in \mathcal{Z}_s, \quad \forall s \\ & & \langle u(r) \rangle_{r \in \mathcal{R}} \in \mathbb{R}^{|\mathcal{R}|}, \\ & \text{s.t.} & z_s(r) = u(r), \quad \forall s, r \in \bar{\mathcal{R}}_s. \end{array} \tag{5}$$

If the score functions f_s are convex, P' becomes a convex program (unlike P, which is discrete); being a relaxation, it provides an upper bound of P.

Lagrangian. Introducing a Lagrange multiplier $\lambda_s(r)$ for each agreement constraint in Eq. 5, one obtains the Lagrangian function

$$\begin{aligned} \mathcal{L}(z, u, \lambda) = & \sum_{s=1}^{S} \left(f_s(z_s) + \sum_{r \in \bar{\mathcal{R}}_s} \lambda_s(r) z_s(r) \right) \\ & - \sum_{r \in \mathcal{R}} \left(\sum_{s: r \in \bar{\mathcal{R}}_s} \lambda_s(r) \right) u(r), \end{aligned} \tag{6}$$

and the dual problem (the *master*)

$$\begin{array}{lll} D: & \text{minimize} & \sum_{s=1}^{S} g_s(\lambda_s) \\ & \text{w.r.t.} & \lambda = \langle \lambda_1, \ldots, \lambda_S \rangle \\ & \text{s.t.} & \sum_{s: r \in \bar{\mathcal{R}}_s} \lambda_s(r) = 0, \ \forall r \in \mathcal{R}, \end{array} \tag{7}$$

where the $g_s(\lambda_s)$ are the solution values of the following subproblems (the *slaves*):

$$\begin{array}{ll} \text{maximize} & f_s(z_s) + \sum_{r \in \bar{\mathcal{R}}_s} \lambda_s(r) z_s(r) \\ \text{w.r.t.} & z_s \in \mathcal{Z}_s. \end{array} \tag{8}$$

We assume that strong duality holds (w.r.t. Eqs. 5–7), hence we have $P \leq P' = D$.[3]

Solving the dual. Why is the dual formulation D (Eqs. 7–8) more appealing than P' (Eq. 5)? The answer is that the components $1, \ldots, S$ are now decoupled, which makes things easier provided each slave subproblem (Eq. 8) can be solved efficiently. In fact, this is always a concern in the mind of the model's designer when she chooses a decomposition (the framework that we describe in §3, in some sense, alleviates her from this concern). If the score functions are linear, *i.e.*, of the form $f_s(z_s) = \sum_{r \in \mathcal{R}_s} \theta_s(r) z_s(r)$ for some vector $\theta_s = \langle \theta_s(r) \rangle_{r \in \mathcal{R}_s}$, then Eq. 8 becomes a linear program, for which a solution exists at a vertex of $\mathcal{Z}_s$ (which

gle factor in a factor graph (Smith and Eisner, 2008), or to a entire subgraph enclosing several factors (Koo et al., 2010), or even to a formula in Markov logic (Richardson and Domingos, 2006). In these examples, the basic parts may correspond to individual variable-value pairs.

[2] Instead of following the path $P \to P' \to D$, Rush et al. (2010) go straight from P to D via a Lagrangian relaxation. The two formulations are equivalent for linear score functions.

[3] This is guaranteed if the score functions f_s are linear.

in turn is an element of $\mathcal{Y}_s$). Depending on the structure of the problem, Eq. 8 may be solved by brute force, dynamic programming, or specialized combinatorial algorithms (Rush et al., 2010; Koo et al., 2010; Rush and Collins, 2011).

Applying the projected subgradient method (Komodakis et al., 2007; Rush et al., 2010) to the master problem (Eq. 7) yields a remarkably simple algorithm, which at each round t solves the subproblems in Eq. 8 for $s = 1, \ldots, S$, and then gathers these solutions (call them z_s^{t+1}) to compute an "averaged" vote for each basic part,

$$u^{t+1}(r) = \frac{1}{\delta(r)} \sum_{s:r \in \bar{\mathcal{R}}_s} z_s^{t+1}(r), \quad (9)$$

where $\delta(r) = |\{s : r \in \mathcal{R}_s\}|$ is the number of components which contain part r. An update of the Lagrange variables follows,

$$\lambda_s^{t+1}(r) = \lambda_s^t(r) - \eta_t(z_s^{t+1}(r) - u^{t+1}(r)), \quad (10)$$

where η_t is a stepsize. Intuitively, the algorithm pushes for a consensus among the slaves (Eq. 9), via an adjustment of the Lagrange multipliers which takes into consideration deviations from the average (Eq. 10). The subgradient method is guaranteed to converge to the solution of D (Eq. 7), for suitably chosen stepsizes (Shor, 1985; Bertsekas et al., 1999); it also provides a certificate of optimality in case the relaxation is tight (*i.e.*, $P = D$) and the *exact* solution has been found. However, convergence is slow when S is large (as we will show in the experimental section), and no certificates are available when there is a relaxation gap ($P < P'$). In the next section, we describe the DD-ADMM algorithm (Martins et al., 2011), which does not have these drawbacks and shares a similar simplicity.

3 Alternating Directions Method

There are two reasons why subgradient-based dual decomposition is not completely satisfying:

- it may take a long time to reach a consensus;
- it puts all its resources in solving the dual problem D, and does not attempt to make progress in the primal P', which is closer to our main concern.[4]

[4]Our main concern is P; however solving P' is often a useful step towards that goal, either because a good rounding scheme exists, or because one may build tighter relaxations to approach P (Sontag et al., 2008; Rush and Collins, 2011).

Taking a look back at the relaxed primal problem P' (Eq. 5), we see that any primal feasible solution must satisfy the agreement constraints. This suggests that penalizing violations of these constraints could speed up consensus.

Augmented Lagrangian. By adding a penalty term to Eq. 6, we obtain the *augmented Lagrangian function* (Hestenes, 1969; Powell, 1969):

$$\mathcal{A}_\rho(z, u, \lambda) = \mathcal{L}(z, u, \lambda) - \frac{\rho}{2} \sum_{s=1}^{S} \sum_{r \in \bar{\mathcal{R}}_s} (z_s(r) - u(r))^2, \quad (11)$$

where the parameter $\rho \geq 0$ controls the intensity of the penalty. Augmented Lagrangian methods are well-known in the optimization community (see, *e.g.*, Bertsekas et al. (1999), §4.2). They alternate updates to the λ-variables, while seeking to maximize $\mathcal{A}_\rho$ with respect to z and u. In our case, however, this joint maximization poses difficulties, since the penalty term couples the two variables. The *alternating directions method of multipliers* (ADMM), coined by Gabay and Mercier (1976) and Glowinski and Marroco (1975), sidesteps this issue by performing alternate maximizations,

$$z^{t+1} = \arg\max_z \mathcal{A}_\rho(z, u^t, \lambda^t), \quad (12)$$

$$u^{t+1} = \arg\max_u \mathcal{A}_\rho(z^{t+1}, u, \lambda^t), \quad (13)$$

followed by an update of the Lagrange multipliers as in Eq. 10. Recently, ADMM has attracted interest, being applied in a variety of problems; see the recent book by Boyd et al. (2011) for an overview. As derived in the App. A, the u-updates in Eq. 13 have a closed form, which is precisely the averaging operation performed by the subgradient method (Eq. 9). We are left with the problem of computing the z-updates. Like in the subgradient approach, the maximization in Eq. 12 can be separated into S independent slave subproblems, which now take the form:

$$\begin{array}{ll} \text{maximize} & f_s(z_s) + \sum_{r \in \bar{\mathcal{R}}_s} \lambda_s(r) z_s(r) \\ & -\frac{\rho}{2} \sum_{r \in \bar{\mathcal{R}}_s} (z_s(r) - u^t(r))^2 \\ \text{w.r.t.} & z_s \in \mathcal{Z}_s(x). \end{array} \quad (14)$$

Comparing Eq. 8 and Eq. 14, we observe that the only difference is the presence in the latter of a

quadratic term which regularizes towards the previous averaged votes $u^t(r)$. Because of this term, the solution of Eq. 14 for linear score functions may not be at a vertex (in contrast to the subgradient method). We devote §4 to describing exact and efficient ways of solving the problem in Eq. 14 for important, widely used slaves. Before going into details, we mention another advantage of ADMM over the subgradient algorithm: it knows when to stop.

Primal and dual residuals. Recall that the subgradient method provides optimality certificates when the relaxation is tight ($P = P'$) and an exact solution of P has been found. While this is good enough when tight relaxations are frequent, as in the settings explored by Rush et al. (2010), Koo et al. (2010), and Rush and Collins (2011), it is hard to know when to stop when a relaxation gap exists. We would like to have similar guarantees concerning the relaxed primal P'.[5] A general weakness of subgradient algorithms is that they do not have this capacity, and so are usually stopped by specifying a maximum number of iterations. In contrast, ADMM allows to keep track of primal and dual residuals (Boyd et al., 2011). This allows providing certificates not only for the exact solution of P (when the relaxation is tight), but also to terminate when a near optimal solution of the relaxed problem P' has been found. The *primal residual* r_P^t measures the amount by which the agreement constraints are violated:

$$r_P^t = \frac{\sum_{s=1}^{S} \sum_{r \in \bar{\mathcal{R}}_s} (z_s^t(r) - u^t(r))^2}{\sum_{r \in \mathcal{R}} \delta(r)}; \qquad (15)$$

the *dual residual* r_D^t is the amount by which a dual optimality condition is violated (see Boyd et al. (2011), p.18, for details). It is computed via:

$$r_D^t = \frac{\sum_{r \in \mathcal{R}} \delta(r)(u^t(r) - u^{t-1}(r))^2}{\sum_{r \in \mathcal{R}} \delta(r)}, \qquad (16)$$

Our stopping criterion is thus that these two residuals are below a threshold, *e.g.*, 1×10^{-3}. The complete algorithm is depicted as Alg. 1. As stated in Martins et al. (2011), convergence to the solution of P' is guaranteed with a fixed stepsize $\eta_t = \tau\rho$, with $\tau \in [1, 1.618]$ (Glowinski and Le Tallec, 1989, Thm. 4.2). In our experiments, we set $\tau = 1.5$, and adapt ρ as described in (Boyd et al., 2011, p.20).[6]

Algorithm 1 ADMM-based Dual Decomposition

input: score functions $\langle f_s(.)\rangle_{s=1}^S$, parameters ρ, η, thresholds ϵ_P and ϵ_D.
initialize $t \leftarrow 1$
initialize $u^1(r) \leftarrow 0.5$ and $\lambda_s^1(r) \leftarrow 0, \quad \forall s, \forall r \in \bar{\mathcal{R}}_s$
repeat
  for each $s = 1, \ldots, S$ **do**
    make a z_s-update, yielding z_s^{t+1} (Eq. 14)
  end for
  make a u-update, yielding u^{t+1} (Eq. 9)
  make a λ-update, yielding λ^{t+1} (Eq. 10)
  $t \leftarrow t + 1$
until $r_P^{t+1} < \epsilon_P$ and $r_D^{t+1} < \epsilon_D$ (Eqs. 15–16)
output: relaxed primal and dual solutions u, z, λ

4 Solving the Subproblems

In this section, we address the slave subproblems of DD-ADMM (Eq. 14). We show how these subproblems can be solved efficiently for several important cases that arise in NLP applications. Throughout, we assume that the score functions f_s are linear, *i.e.*, they can be written as $f_s(z_s) = \sum_{r \in \mathcal{R}_s} \theta_s(r) z_s(r)$. This is the case whenever a linear model is used, in which case $\theta_s(r) = \frac{1}{\delta(r)} \boldsymbol{w} \cdot \boldsymbol{\phi}(x, r)$, where $\boldsymbol{w}$ is a weight vector and $\boldsymbol{\phi}(x, r)$ is a feature vector. It is also the scenario studied in previous work in dual decomposition (Rush et al., 2010). Under this assumption, and discarding constant terms, the slave subproblem in Eq. 14 becomes:

$$\max_{z_s \in \mathcal{Z}_s} \sum_{r \in \mathcal{R}_s \setminus \bar{\mathcal{R}}_s} \theta_s(r) z_s(r) - \frac{\rho}{2} \sum_{r \in \bar{\mathcal{R}}_s} (z_s(r) - a_s(r))^2. \qquad (17)$$

where $a_s(r) = u^t(r) + \rho^{-1}(\theta_s(r) + \lambda_s^t(r))$. Since $\mathcal{Z}_s$ is a polytope, Eq. 17 is a quadratic program, which can be solved with a general purpose solver. However, that does not exploit the structure of $\mathcal{Z}_s$ and is inefficient when $|\mathcal{R}_s|$ is large. We next show that for many cases, a closed-form solution is available and

[5] This problem is more important than it may look. Problems with many slaves tend to be less exact, hence relaxation gaps are frequent. Also, when decoding is embedded in training, it is useful to obtain the *fractional* solution of the relaxed primal P (rather than an approximate integer solution). See Kulesza and Pereira (2007) and Martins et al. (2009b) for details.

[6] Briefly, we initialize $\rho = 0.03$ and then increase/decrease ρ by a factor of 2 whenever the primal residual becomes > 10 times larger/smaller than the dual residual.

can be computed in $O(|\mathcal{R}_s|)$ time, up to log factors.[7]

Pairwise Factors. This is the case where $\mathcal{R}_{\text{PAIR}} = \{r_1, r_2, r_{12}\}$, where r_1 and r_2 are basic parts and r_{12} is their conjunction, *i.e.*, we have $\mathcal{Y}_{\text{PAIR}} = \{\langle z_1, z_2, z_{12}\rangle \mid z_{12} = z_1 \wedge z_2\}$. This factor is useful to make conjunctions of variables participate in the score function (see *e.g.* the grandparent, sibling, and head bigram parts in Fig. 1). The convex hull of $\mathcal{Y}_{\text{PAIR}}$ is the polytope $\mathcal{Z}_{\text{PAIR}} = \{\langle z_1, z_2, z_{12}\rangle \in [0,1]^3 \mid z_{12} \leq z_1,\ z_{12} \leq z_2, z_{12} \geq z_1 + z_2 - 1\}$, as shown by Martins et al. (2010). In this case, problem (17) can be written as

$$\begin{array}{ll} \max & \theta_{12} z_{12} - \frac{\rho}{2}[(z_1 - a_1)^2 + (z_2 - a_2)^2] \\ \text{w.r.t.} & \langle z_1, z_2, z_{12}\rangle \in [0,1]^3 \\ \text{s.t.} & z_{12} \leq z_1,\ z_{12} \leq z_2,\ z_{12} \geq z_1 + z_2 - 1 \end{array} \tag{18}$$

and has a closed form solution (see App. B).

Uniqueness Quantification and XOR. Many problems involve constraining variables to take a single value: for example, in dependency parsing, a modifier can only take one head. This can be expressed as the statement $\exists! y : Q(y)$ in first-order logic,[8] or as a one-hot XOR factor in a factor graph (Smith and Eisner, 2008; Martins et al., 2010). In this case, $\mathcal{R}_{\text{XOR}} = \{r_1, \ldots, r_n\}$, and $\mathcal{Y}_{\text{XOR}} = \{\langle z_1, \ldots, z_n\rangle \in \{0,1\}^n \mid \sum_{i=1}^n z_i = 1\}$. The convex hull of $\mathcal{Y}_{\text{XOR}}$ is $\mathcal{Z}_{\text{XOR}} = \{\langle z_1, \ldots, z_n\rangle \in [0,1]^n \mid \sum_{i=1}^n z_i = 1\}$. Assume for the sake of simplicity that all parts in $\mathcal{R}_{\text{XOR}}$ are basic.[9] Up to a constant, the slave subproblem becomes:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\sum_{i=1}^n (z_i - a_i)^2 \\ \text{w.r.t.} & \langle z_1, \ldots, z_n\rangle \in [0,1]^n \\ \text{s.t.} & \sum_i^n z_i = 1. \end{array} \tag{19}$$

This is the problem of projecting onto the probability simplex, which can be done in $O(n \log n)$ time via a sort operation (see App. C).[10]

[7] This matches the asymptotic time that would be necessary to solve the corresponding problems in the subgradient method, for which algorithms are straightforward to derive. The point is that with ADMM fewer instances of these subproblems need to be solved, due to faster convergence of the master problem.

[8] The symbol $\exists!$ means "there is one and only one."

[9] A similar derivation can be made otherwise.

[10] Also common is the need for constraining existence of "at most one" element. This can be reduced to uniqueness quantification by adding a dummy NULL label.

Existential Quantification and OR. Sometimes, only existence is required, not necessarily uniqueness. This can be expressed with disjunctions, existential quantifiers in first-order logic ($\exists y : Q(y)$), or as a OR factor. In this case, $\mathcal{R}_{\text{OR}} = \{r_1, \ldots, r_n\}$, $\mathcal{Y}_{\text{OR}} = \{\langle z_1, \ldots, z_n\rangle \in \{0,1\}^n \mid \bigvee_{i=1}^n z_i = 1\}$, and the convex hull is $\mathcal{Z}_{\text{OR}} = \{\langle z_1, \ldots, z_n\rangle \in [0,1]^n \mid \sum_{i=1}^n z_i \geq 1\}$ (see Tab. 1 in Martins et al. (2010)). The slave subproblem becomes:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\sum_{i=1}^n (z_i - a_i)^2 \\ \text{w.r.t.} & \langle z_1, \ldots, z_n\rangle \in [0,1]^n \\ \text{s.t.} & \sum_i^n z_i \geq 1. \end{array} \tag{20}$$

We derive a procedure in App. D to compute this projection in $O(n \log n)$ runtime, also with a sort.

Negations. The two cases above can be extended to allow some of their inputs to be *negated*. By a change of variables in Eqs. 19–20 it is possible to reuse the same black box that solves those problems. The procedure is as follows:

1. For $i = 1, \ldots, n$, set $a'_i = 1 - a_i$ if the ith variable is negated, and $a'_i = a_i$ otherwise.
2. Obtain $\langle z'_1, \ldots, z'_n\rangle$ as the solution of Eqs. 19 or 20 providing $\langle a'_1, \ldots, a'_n\rangle$ as input.
3. For $i = 1, \ldots, n$, set $z_i = 1 - z'_i$ if the ith variable is negated, and $z_i = z'_i$ otherwise.

The ability to handle negated variables adds a great degree of flexibility. From De Morgan's laws, we can now handle conjunctions and implications (since $\bigwedge_{i=1}^n Q_i(x) \Rightarrow R(x)$ is equivalent to $\bigvee_{i=1}^n \neg Q_i(x) \vee R(x)$).

Logical Variable Assignments. All previous examples involve taking a group of existing variables and defining a constraint. Alternatively, we may want to define a new variable which is the result of an operation involving other variables. For example, $R(x) := \exists! y : Q(x, y)$. This corresponds to the XOR-WITH-OUTPUT factor in Martins et al. (2010). Interestingly, this can be expressed as a XOR where $R(x)$ is negated (*i.e.*, either $\neg R(x)$ holds or exactly one y satisfies $Q(x, y)$, but not both).

A more difficult problem is that of the OR-WITH-OUTPUT factor, expressed by the formula $R(x) := \exists y : Q(x, y)$. We have $\mathcal{R}_{\text{OR-OUT}} = \{r_0, \ldots, r_n\}$, and $\mathcal{Y}_{\text{OR-OUT}} = \{\langle z_0, \ldots, z_n\rangle \in \{0,1\}^n \mid z_0 =$

		# Slaves	Runtime	Description
Tree	$\exists! h : \texttt{arc}(h, m), \quad m \neq 0$	$O(n)$	$O(n \log n)$	Each non-root word has a head
	$\texttt{flow}(h, m, k) \Rightarrow \texttt{arc}(h, m)$	$O(n^3)$	$O(1)$	Only active arcs may carry flow
	$\texttt{path}(m, d) := \exists! h : \texttt{flow}(h, m, d), \quad m \neq 0$	$O(n^2)$	$O(n \log n)$	
	$\texttt{path}(h, d) := \exists! m : \texttt{flow}(h, m, d)$	$O(n^2)$	$O(n \log n)$	Paths and flows are consistent
	$\texttt{path}(0, m) := \text{TRUE}, \quad \texttt{flow}(h, m, m) := \text{TRUE}$			(see Martins et al. (2010))
All siblings	$\texttt{sibl}(h, m, s) := \texttt{arc}(h, m) \wedge \texttt{arc}(h, s)$	$O(n^3)$	$O(1)$	By definition
Grandp.	$\texttt{grand}(g, h, m) := \texttt{arc}(g, h) \wedge \texttt{arc}(h, m)$	$O(n^3)$	$O(1)$	By definition
Head Bigram	$\texttt{bigram}(b, h, m) := \texttt{arc}(b, m-1) \wedge \texttt{arc}(h, m), \quad m \neq 0$	$O(n^3)$	$O(1)$	By definition
Consec. Sibl.	$\texttt{lastsibl}(h, m, m) := \texttt{arc}(h, m)$			
	$\exists! m \in [h, k] : \texttt{lastsibl}(h, m, k)$	$O(n^2)$	$O(n \log n)$	Head automaton model
	$\texttt{lastsibl}(h, m, k) := \texttt{lastsibl}(h, m, k+1)$			(see supplementary material)
	$\quad \oplus \texttt{nextsibl}(h, m, k+1)$	$O(n^3)$	$O(1)$	
	$\texttt{arc}(h, m) := \exists! s \in [h, m] : \texttt{nextsibl}(h, s, m)$	$O(n^2)$	$O(n \log n)$	
Nonproj. Arc	$\texttt{nonproj}(h, m) := \texttt{arc}(h, m) \wedge \exists k \in [h, m] : \neg \texttt{path}(h, k)$	$O(n^2)$	$O(n \log n)$	By definition

Table 1: First-order logic formulae underlying our dependency parser. The basic parts are the predicate variables $\texttt{arc}(h, m)$ (indicating an arc linking head h to modifier m), $\texttt{path}(a, d)$ (indicating a directed path from ancestor a to descendant d), $\texttt{nextsibl}(h, m, s)$ (indicating that $\langle h, m \rangle$ and $\langle h, s \rangle$ are consecutive siblings), $\texttt{nonproj}(h, m)$ (indicating that $\langle h, m \rangle$ is a non-projective arc), as well as the auxiliary variables $\texttt{flow}(h, m, d)$ (indicating that arc $\langle h, m \rangle$ carries flow to d), and $\texttt{lastsibl}(h, m, k)$ (indicating that, up to position k, the last seen modifier of h occurred at position m). The non-basic parts are the pairwise factors $\texttt{sibl}(h, m, s)$, $\texttt{grand}(g, h, m)$, and $\texttt{bigram}(b, h, m)$; as well as each logical formula. Columns 3–4 indicate the number of parts of each kind, and the time complexity for solving each subproblem. For a sentence of length n, there are $O(n^3)$ parts and the total complexity is $O(n^3 \log n)$.

$\bigvee_{i=1}^n z_i\}$. The convex hull of $\mathcal{Y}_{\text{OR-OUT}}$ is the following set: $\mathcal{Z}_{\text{OR-OUT}} = \{\langle z_0, \ldots, z_n \rangle \in [0, 1]^n \mid z_0 \geq \sum_{i=1}^n z_i, z_0 \leq z_i, \ \forall i = 1, \ldots, n\}$ (Martins et al., 2010, Tab.1). The slave subproblem is:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} \sum_{i=0}^n (z_i - a_i)^2 \\ \text{w.r.t.} & \langle z_0, \ldots, z_n \rangle \in [0, 1]^n \\ \text{s.t.} & z_0 \geq \sum_{i=1}^n z_i; \ z_0 \leq z_i, \ \forall i = 1, \ldots, n. \end{array} \tag{21}$$

The problem in Eq. 21 is more involved than the ones in Eqs. 19–20. Yet, there is still an efficient procedure with runtime $O(n \log n)$ (see App. E). By using the result above for negated variables, we are now endowed with a procedure for many other cases, such that AND-WITH-OUTPUT and formulas with universal quantifiers (*e.g.*, $R(x) := \forall y : Q(x, y)$). Up to a log-factor, the runtimes will be linear in the number of predicates.

Larger Slaves. The only disadvantage of DD-ADMM in comparison with the subgradient algorithm is that there is not an obvious way of solving the subproblem in Eq. 14 exactly for large combinatorial factors, such as the TREE constraint in dependency parsing, or a sequence model. Hence, our method seems to be more suitable for decompositions which involve "simple slaves," even if their number is large. However, this does not rule out the possibility of using this method otherwise. Eckstein and Bertsekas (1992) show that the ADMM algorithm may still converge when the z-updates are inexact. Hence the method may still work if the slaves are solved numerically up to some accuracy. We defer this to future investigation.

5 Experiments: Dependency Parsing

We used 14 datasets with non-projective dependencies from the CoNLL-2006 and CoNLL-2008 shared tasks (Buchholz and Marsi, 2006; Surdeanu et al., 2008). We also used a projective English dataset derived from the Penn Treebank by applying the standard head rules of Yamada and Matsumoto (2003).[11] We did not force the parser to output projective trees or unique roots for any of the datasets; everything is learned from the data. We trained by running 10 iterations of the cost-augmented MIRA algorithm (Crammer et al., 2006) with LP-relaxed decoding, as in Martins et al. (2009b). Following common practice (Charniak and Johnson, 2005; Carreras et al., 2008), we employed a coarse-to-fine procedure to prune away unlikely candidate arcs, as described by Koo and Collins (2010). To ensure valid parse trees at test time, we rounded fractional

[11] As usual, we train on sections §02–21, use §22 as validation data, and test on §23. We ran SVMTool (Giménez and Marquez, 2004) to obtain automatic part-of-speech tags for §22–23.

solutions as described in Martins et al. (2009a) (yet, solutions were integral most of the time).

The parts used in our full model are the ones depicted in Fig. 1. Note that a subgradient-based method could handle some of those parts efficiently (*arcs*, *consecutive siblings*, *grandparents*, and *head bigrams*) by composing arc-factored models, head automata, and a sequence labeler. However, no lightweight decomposition seems possible for incorporating parts for *all siblings*, *directed paths*, and *non-projective arcs*. Tab. 1 shows the first-order logical formulae that encode the constraints in our model. Each formula gives rise to a subproblem which is efficiently solvable (see §4). By ablating some of rows of Tab. 1 we recover known methods:

- Resorting to the *tree* and *consecutive sibling* formulae gives one of the models in Koo et al. (2010), with the *same* linear relaxation (a proof of this fact is included in App. F);
- Resorting to *tree*, *all siblings*, *grandparent*, and *non-projective arcs*, recovers a multi-commodity flow configuration proposed by Martins et al. (2009a); the relaxation is also the same.[12]

The experimental results are shown in Tab. 2. For comparison, we include the best published results for each dataset (at the best of our knowledge), among transition-based parsers (Nivre et al., 2006; Huang and Sagae, 2010), graph-based parsers (McDonald et al., 2006; Koo and Collins, 2010), hybrid methods (Nivre and McDonald, 2008; Martins et al., 2008), and turbo parsers (Martins et al., 2010; Koo et al., 2010). Our full model achieved the best reported scores for 7 datasets. The last two columns show a consistent improvement (with the exceptions of Chinese and Arabic) when using the full set of features over a second order model with grandparent and consecutive siblings, which is our reproduction of the model of Koo et al. (2010).[13]

[12] Although Martins et al. (2009a) also incorporated consecutive siblings in one of their configurations, our constraints are tighter than theirs. See App. F.

[13] Note however that the actual results of Koo et al. (2010) are higher than our reproduction, as can be seen in the second column. The differences are due to the features that were used and on the way the models were trained. The cause is not search error: exact decoding with an ILP solver (CPLEX) revealed no significant difference with respect to our G+CS column. We leave further analysis for future work.

	Best known UAS		G+CS	Full
Arabic	80.18	[Ma08]	**81.12**	81.10 (-0.02)
Bulgar.	92.88	[Ma10]	93.04	**93.50** (+0.46)
Chinese	**91.89**	[Ma10]	91.05	90.62 (-0.43)
Czech	88.78	[Ma10]	88.80	**89.46** (+0.66)
English	92.57	[Ko10]	92.45	**92.68** (+0.23)
Danish	91.78	[Ko10]	91.70	**91.86** (+0.16)
Dutch	**85.81**	[Ko10]	84.77	85.53 (+0.76)
German	91.49	[Ma10]	91.29	**91.89** (+0.60)
Japane.	93.42	[Ma10]	93.62	**93.72** (+0.10)
Portug.	**93.03**	[Ko10]	92.05	92.29 (+0.24)
Slovene	86.21	[Ko10]	86.09	**86.95** (+0.86)
Spanish	**87.04**	[Ma10]	85.99	86.74 (+0.75)
Swedish	**91.36**	[Ko10]	89.94	90.16 (+0.22)
Turkish	**77.55**	[Ko10]	76.24	76.64 (+0.40)
PTB §23	**93.04**	[KC10]	92.19	92.53 (+0.34)

Table 2: Unlabeled attachment scores, excluding punctuation. In the second column, [Ma08] denotes Martins et al. (2008), [KC10] is Koo and Collins (2010), [Ma10] is Martins et al. (2010), and [Ko10] is Koo et al. (2010). In columns 3–4, "Full" is our full model, and "G+CS" is our reproduction of the model of Koo et al. (2010), *i.e.*, the same as "Full" but with all features ablated excepted for grandparents and consecutive siblings.

	AF	+G+CS	+AS	+NP	Full
PTB §22	91.02	92.13	92.32	92.36	92.41
PTB §23	91.36	92.19	92.41	92.50	92.53

Table 3: Feature ablation experiments. AF is an arc-factored model; +G+CS adds grandparent and consecutive siblings; +AS adds all-siblings; +NP adds non-projective arcs; Full adds the bigram and directed paths.

Feature ablation and error analysis. We conducted a simple ablation study by training several models on the English PTB with different sets of features. Tab. 3 shows the results. As expected, performance keeps increasing as we use models with greater expressive power. We show some concrete examples in App. G of sentences that the full model parsed correctly, unlike less expressive models.

Convergence speed and optimality. Fig. 2 compares the performance of DD-ADMM and the subgradient algorithms in the validation section of the PTB.[14] For the second order model, the subgradient

[14] The learning rate in the subgradient method was set as $\eta_t = \eta_0/(1+N_{\text{incr}}(t))$, as in Koo et al. (2010), where $N_{\text{incr}}(t)$ is the number of dual increases up to the tth iteration, and η_0 is chosen to maximize dual decrease after 20 iterations (in a per sentence basis). Those preliminary iterations are not plotted in Fig. 2.

method has more slaves than in Koo et al. (2010): it has a slave imposing the TREE constraint (whose subproblems consists on finding a minimum spanning tree) and several for the all-sibling parts, yielding an average number of 310.5 and a maximum of 4310 slaves. These numbers are still manageable, and we observe that a "good" UAS is achieved relatively quickly. The ADMM method has many more slaves due to the multicommodity flow constraints (average 1870.8, maximum 65446), yet it attains optimality sooner, as can be observed in the right plot. For the full model, the subgradient-based method becomes extremely slow, and the UAS score severely degrades (after 1000 iterations it is 2% less than the one obtained with the ADMM-based method, with very few instances having been solved to optimality). The reason is the number of slaves: in this configuration and dataset the average number of slaves per instance is 3327.4, and the largest number is 113207. On the contrary, the ADMM method keeps a robust performance, with a large fraction of optimality certificates in early iterations.

Runtime and caching strategies. Despite its suitability to problems with many overlapping components, our parser is still 1.6 times slower than Koo et al. (2010) (0.34 against 0.21 sec./sent. in PTB §23), and is far beyond the speed of transition-based parsers (*e.g.*, Huang and Sagae (2010) take 0.04 sec./sent. on the same data, although accuracy is lower, 92.1%). Our implementation, however, is not fully optimized. We next describe how considerable speed-ups are achieved by caching the subproblems, following a strategy similar to Koo et al. (2010).

Fig. 3 illustrates the point. After a few iterations, many variables $u(r)$ see a consensus being achieved (*i.e.*, $u^t(r) = z_s^{t+1}(r), \forall s$) and enter an idle state: they are left unchanged by the u-update in Eq. 9, and so do the Lagrange variables $\lambda_s^{t+1}(r)$ (Eq. 10). If by iteration t all variables in a subproblem s are idle, then $z_s^{t+1}(r) = z_s^t(r)$, hence the subproblem does not need to be resolved.[15] Fig. 3 shows that many variables and subproblems are left untouched after the first few rounds.

[15]Even if not all variables are idle in s, caching may still be useful: note that the z-updates in Eq. 14 tend to be sparse for the subproblems described in §4 (these are Euclidean projections onto polytopes with 0/1 vertices, which tend to hit corners). Another trick that may accelerate the algorithm is warm-starting: since many subproblems involve a sort operation, storing the sorted indexes may speedup the next round.

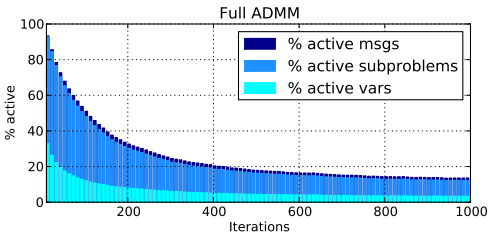


Figure 3: Fraction of active variables, subproblems and messages along DD-ADMM iterations (full model). The number of active messages denotes the total number of variables (active or not) that participate in an active factor.

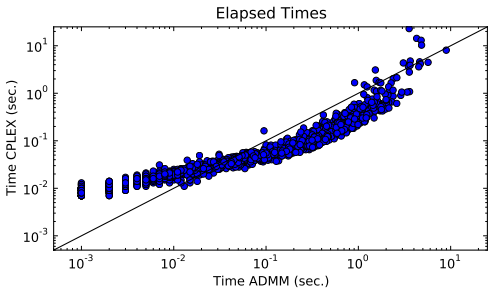


Figure 4: Runtimes of DD-ADMM and CPLEX on PTB §22 (each point is a sentence). Average runtimes are 0.362 (DD-ADMM) and 0.565 sec./sent. (CPLEX).

Finally, Fig. 4 compares the runtimes of our implementation of DD-ADMM with those achieved by a state-of-the-art LP solver, CPLEX, in its best performing configuration: the simplex algorithm applied to the dual LP. We observe that DD-ADMM is faster in some regimes but slower in others. For short sentences (< 15 words), DD-ADMM tends to be faster. For longer sentences, CPLEX is quite effective as it uses good heuristics for the pivot steps in the simplex algorithm; however, we observed that it sometimes gets trapped on large problems. Note also that DD-ADMM is not fully optimized, and that it is much more amenable to parallelization than the simplex algorithm, since it is composed of many independent slaves. This suggests potentially significant speed-ups in multi-core environments.

6 Related Work

Riedel and Clarke (2006) first formulated dependency parsing as an integer program, along with logical constraints. The multicommodity flow for-

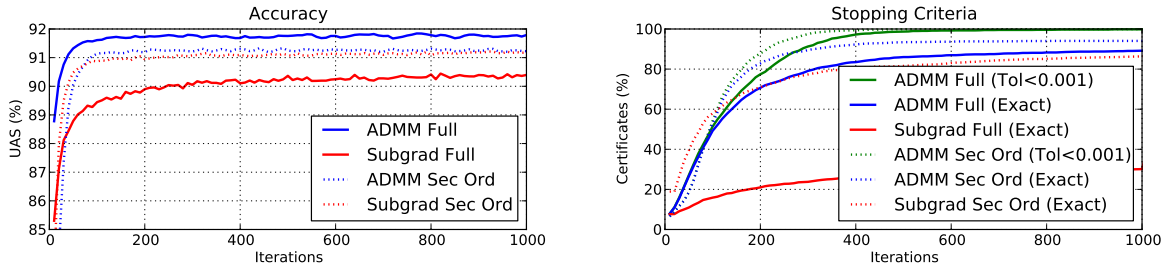


Figure 2: UAS including punctuation (left) and fraction of optimality certificates (right) accross iterations of the subgradient and DD-ADMM algorithms, in PTB §22. "Full" is our full model; "Sec Ord" is a second-order model with grandparents and all siblings, for which the subgradient method uses a coarser decomposition with a TREE factor. Since subgradient and DD-ADMM are solving the same problems, the solid lines (as the dashed ones) would meet in the limit, however subgradient converges very slowly for the full model. The right plot shows optimality certificates for both methods, indicating that an exact solution of P has been found; for DD-ADMM we also plot the fraction of instances that converged to an accurate solution of P' (primal and dual residuals $< 10^{-3}$) and hence can be stopped.

mulation was introduced by Martins et al. (2009a), along with some of the parts considered here. Koo et al. (2010) proposed a subgradient-based dual decomposition method that elegantly combines head automata with maximum spanning tree algorithms; these parsers, as well as the loopy belief propagation method of Smith and Eisner (2008), are all instances of turbo parsers (Martins et al., 2010).

DD-ADMM has been proposed and theoretically analyzed by Martins et al. (2011) for problems representable as factor graphs. The general ADMM method has a long-standing history in optimization (Hestenes, 1969; Powell, 1969; Glowinski and Marroco, 1975; Gabay and Mercier, 1976; Boyd et al., 2011). Other methods have been recently proposed to accelerate dual decomposition, such as Jojic et al. (2010) and Meshi and Globerson (2011) (the latter applying ADMM in the dual rather than the primal).

While our paper shows limitations of the subgradient method when there are many overlapping components, this method may still be advantageous over ADMM in problems that are nicely decomposable, since it often allows reusing existing combinatorial machinery. Yet, the scenario we consider here is realistic in NLP, where we often have to deal with not-lightly-decomposable constrained problems (e.g., exploiting linguistic knowledge).

7 Conclusion

We have introduced new feature-rich turbo parsers. Since exact decoding is intractable, we solve an LP relaxation through a recently proposed consensus algorithm, DD-ADMM, which is suitable for problems with many overlapping components. We study the empirical runtime and convergence properties of DD-ADMM, complementing the theoretical treatment in Martins et al. (2011). DD-ADMM compares favourably against the subgradient method in several aspects: it is faster to reach a consensus, it has better stopping conditions, and it works better in non-lightweight decompositions. While its slave subproblems are more involved, we derived closed-form solutions for many cases of interest, such as first-order logic formulas and combinatorial factors.

DD-ADMM may be useful in other frameworks involving logical constraints, such as the models for compositional semantics presented by Liang et al. (2011). Non-logical constraints may also yield efficient subproblems, *e.g.*, the length constraints in summarization and compression (Clarke and Lapata, 2008; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011). Finally, DD-ADMM can be adapted to tighten its relaxations towards exact decoding, as in Sontag et al. (2008) and Rush and Collins (2011). We defer this for future work.

Acknowledgments

We thank all reviewers for their comments, Eric Xing for helpful discussions, and Terry Koo and Sasha Rush for answering questions about their parser and for providing code. A. M. was supported by a FCT/ICTI grant through the CMU-Portugal Program, and by Priberam. This work was partially supported by the FET programme (EU FP7), under the SIMBAD project (contract 213250). N. S. was supported by NSF CAREER IIS-1054319.

References

M. Auli and A. Lopez. 2011. A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing. In *Proc. of ACL*.

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

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proc. of ACL*.

D. Bertsekas, W. Hager, and O. Mangasarian. 1999. *Nonlinear programming*. Athena Scientific.

D.P. Bertsekas, A. Nedic, and A.E. Ozdaglar. 2003. *Convex analysis and optimization*. Athena Scientific.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Publishers (to appear).

J.P. Boyle and R.L. Dykstra. 1986. A method for finding projections onto the intersections of convex sets in Hilbert spaces. In *Advances in order restricted statistical inference*, pages 28–47. Springer Verlag.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

X. Carreras, M. Collins, and T. Koo. 2008. TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-rich Parsing. In *CONLL*.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *CoNLL*.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. ACL*, pages 173–180. Association for Computational Linguistics Morristown, NJ, USA.

D. Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

J. Clarke and M. Lapata. 2008. Global Inference for Sentence Compression An Integer Linear Programming Approach. *JAIR*, 31:399–429.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online Passive-Aggressive Algorithms. *JMLR*, 7:551–585.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. 2008. Efficient projections onto the L1-ball for learning in high dimensions. In *ICML*.

J. Eckstein and D. Bertsekas. 1992. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318.

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

D. Gabay and B. Mercier. 1976. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40.

J. Giménez and L. Marquez. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *Proc. of LREC*.

R. Glowinski and P. Le Tallec. 1989. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*. Society for Industrial Mathematics.

R. Glowinski and A. Marroco. 1975. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par penalisation-dualité, d'une classe de problèmes de Dirichlet non linéaires. *Rev. Franc. Automat. Inform. Rech. Operat.*, 9:41–76.

M. Hestenes. 1969. Multiplier and gradient methods. *Jour. Optim. Theory and Applic.*, 4:302–320.

L. Huang and K. Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proc. of ACL*, pages 1077–1086.

R. Johansson and P. Nugues. 2008. Dependency-based Semantic Role Labeling of PropBank. In *EMNLP*.

V. Jojic, S. Gould, and D. Koller. 2010. Accelerated dual decomposition for MAP inference. In *ICML*.

N. Komodakis, N. Paragios, and G. Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*, pages 1–11.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *EMNLP*.

A. Kulesza and F. Pereira. 2007. Structured Learning with Approximate Inference. *NIPS*.

P. Liang, M.I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Proc. Association for Computational Linguistics (ACL)*.

A. F. T. Martins and N. A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *NAACL-HLT Workshop on Integer Linear Programming for NLP*.

A. F. T. Martins, D. Das, N. A. Smith, and E. P. Xing. 2008. Stacking dependency parsers. In *EMNLP*.

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

A. F. T. Martins, N. A. Smith, and E. P. Xing. 2009b. Polyhedral outer approximations with application to natural language parsing. In *ICML*.

A. F. T. Martins, N. A. Smith, E. P. Xing, M. A. T. Figueiredo, and P. M. Q. Aguiar. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *EMNLP*.

A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. 2011. An Augmented Lagrangian Approach to Constrained MAP Inference. In *ICML*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *CoNLL*.

O. Meshi and A. Globerson. 2011. An Alternating Direction Method for Dual MAP LP Relaxation. In *ECML PKDD*.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *ACL-HLT*.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Procs. of CoNLL*.

S. Petrov and D. Klein. 2008. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proc. of EMNLP*.

M. Powell. 1969. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1):107–136.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP*.

A. M. Rush and M. Collins. 2011. Exact decoding of syntactic translation models through lagrangian relaxation. In *ACL*.

A. Rush, D. Sontag, M. Collins, and T. Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *EMNLP*.

N. Shor. 1985. *Minimization methods for non-differentiable functions*. Springer.

D. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *EMNLP*.

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

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *CoNLL*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *NIPS*.

R.W. Tromble and J. Eisner. 2006. A fast finite-state relaxation method for enforcing global constraints on sequence decoding. In *Proc. of NAACL*, pages 423–430.

M. Wainwright and M. Jordan. 2008. *Graphical Models, Exponential Families, and Variational Inference.* Now Publishers.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *IWPT*.

Approximate Scalable Bounded Space Sketch for Large Data NLP

Amit Goyal and Hal Daumé III
Dept. of Computer Science
University of Maryland
College Park, MD 20742
{amit,hal}@umiacs.umd.edu

Abstract

We exploit sketch techniques, especially the Count-Min sketch, a memory, and time efficient framework which approximates the frequency of a word pair in the corpus without explicitly storing the word pair itself. These methods use hashing to deal with massive amounts of streaming text. We apply Count-Min sketch to approximate word pair counts and exhibit their effectiveness on three important NLP tasks. Our experiments demonstrate that on all of the three tasks, we get performance comparable to Exact word pair counts setting and state-of-the-art system. Our method scales to 49 GB of unzipped web data using bounded space of 2 billion counters (8 GB memory).

1 Introduction

There is more data available today on the web than there has ever been and it keeps increasing. Use of large data in the Natural Language Processing (NLP) community is not new. Many NLP problems (Brants et al., 2007; Turney, 2008; Ravichandran et al., 2005) have benefited from having large amounts of data. However, processing large amounts of data is still challenging.

This has motivated NLP community to use commodity clusters. For example, Brants et al. (2007) used 1500 machines for a day to compute the relative frequencies of n-grams from 1.8TB of web data. In another work, a corpus of roughly 1.6 Terawords was used by Agirre et al. (2009) to compute pairwise similarities of the words in the test sets using the MapReduce infrastructure on $2,000$ cores. However, the inaccessibility of clusters to an average user has attracted the NLP community to use streaming, randomized, and approximate algorithms to handle large amounts of data (Goyal et al., 2009; Levenberg et al., 2010; Van Durme and Lall, 2010).

Streaming approaches (Muthukrishnan, 2005) provide memory and time-efficient framework to deal with terabytes of data. However, these approaches are proposed to solve a singe problem. For example, our earlier work (Goyal et al., 2009) and Levenberg and Osborne (2009) build approximate language models and show their effectiveness in Statistical Machine Translation (SMT). Stream-based translation models (Levenberg et al., 2010) has been shown effective to handle large parallel streaming data for SMT. In Van Durme and Lall (2009b), a Talbot Osborne Morris Bloom (TOMB) Counter (Van Durme and Lall, 2009a) was used to find the top-K verbs "y" given verb "x" using the highest approximate online Pointwise Mutual Information (PMI) values.

In this paper, we explore sketch techniques, especially the Count-Min sketch (Cormode and Muthukrishnan, 2004) to build a *single* model to show its effectiveness on three important NLP tasks:

- Predicting the Semantic Orientation of words (Turney and Littman, 2003)
- Distributional Approaches for word similarity (Agirre et al., 2009)
- Unsupervised Dependency Parsing (Cohen and Smith, 2010) with a little linguistics knowledge.

In all these tasks, we need to compute association measures like Pointwise Mutual Information (PMI),

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 250–261,
Edinburgh, Scotland, UK, July 27–31, 2011.

and Log Likelihood ratio (LLR) between words. To compute association scores (AS), we need to count the number of times pair of words appear together within a certain window size. However, explicitly storing the counts of all word pairs is both computationally expensive and memory intensive (Agirre et al., 2009; Pantel et al., 2009). Moreover, the memory usage keeps increasing with increase in corpus size.

We explore Count-Min (CM) sketch to address the issue of *efficient storage* of such data. The CM sketch stores counts of all word pairs within a *bounded space*. Storage space saving is achieved by *approximating* the frequency of word pairs in the corpus without explicitly storing the word pairs themselves. Both updating (adding a new word pair or increasing the frequency of existing word pair) and querying (finding the frequency of a given word pair) are constant time operations making it efficient online storage data structure for large data. Sketches are *scalable* and can easily be implemented in distributed setting.

We use CM sketch to store counts of word pairs (except word pairs involving stop words) within a window of size[1] 7 over different size corpora. We store exact counts of words (except stop words) in hash table (since the number of unique words is not large that is quadratically less than the number of unique word pairs). The approximate PMI and LLR scores are computed using these approximate counts and are applied to solve our three NLP tasks. Our experiments demonstrate that on all of the three tasks, we get performance comparable to Exact word pair counts setting and state-of-the-art system. Our method scales to 49 GB of unzipped web data using bounded space of 2 billion counters (8 GB memory). This work expands upon our earlier workshop papers (Goyal et al., 2010a; Goyal et al., 2010b).

2 Sketch Techniques

A sketch is a compact summary data structure to store the frequencies of all items in the input stream. Sketching techniques use hashing to map items in streaming data onto a small sketch vector that can be updated and queried in constant time. These techniques generally process the input stream in one direction, say from left to right, without re-processing previous input. The main advantage of using these techniques is that they require a storage which is sub-linear in size of the input stream. The following surveys comprehensively review the streaming literature: (Rusu and Dobra, 2007; Cormode and Hadjieleftheriou, 2008).

There exists an extensive literature on sketch techniques (Charikar et al., 2004; Li et al., 2008; Cormode and Muthukrishnan, 2004; Rusu and Dobra, 2007) in algorithms community for solving many large scale problems. However, in practice, researchers have preferred Count-Min (CM) sketch over other sketch techniques in many application areas, such as Security (Schechter et al., 2010), Machine Learning (Shi et al., 2009; Aggarwal and Yu, 2010), and Privacy (Dwork et al., 2010). This motivated us to explore CM sketch to solve three important NLP problems.[2]

2.1 Count-Min Sketch

The Count-Min sketch (Cormode and Muthukrishnan, 2004) is a compact summary data structure used to store the frequencies of all items in the input stream. The sketch allows fundamental queries on the data stream such as point, range and inner product queries to be approximately answered very quickly. It can also be applied to solve the finding frequent items problem (Manku and Motwani, 2002) in a data stream. In this paper, we are only interested in point queries. The aim of a point query is to estimate the count of an item in the input stream. For other details, the reader is referred to (Cormode and Muthukrishnan, 2004).

Given an input stream of word pairs of length N and user chosen parameters δ and ϵ, the algorithm stores the frequencies of all the word pairs with the following guarantees:

- All reported frequencies are within the true frequencies by at most ϵN with a probability of at least 1-δ.
- The space used by the algorithm is $O(\frac{1}{\epsilon}\log\frac{1}{\delta})$.

[1] 7 is chosen from intuition and not tuned.

[2] In future, in another line of research, we will explore comparing different sketch techniques for NLP problems.

- Constant time of O($\log(\frac{1}{\delta})$) per each update and query operation.

2.1.1 CM Data Structure

A Count-Min sketch (CM) with parameters (ϵ,δ) is represented by a two-dimensional array with width w and depth d :

$$\begin{bmatrix} \text{sketch}[1,1] & \cdots & \text{sketch}[1,w] \\ \vdots & \ddots & \vdots \\ \text{sketch}[d,1] & \cdots & \text{sketch}[d,w] \end{bmatrix}$$

Among the user chosen parameters, ϵ controls the amount of tolerable error in the returned count and δ controls the probability with which the returned count is not within the accepted error. These values of ϵ and δ determine the width and depth of the two-dimensional array respectively. To achieve the guarantees mentioned in the previous section, we set w=$\frac{2}{\epsilon}$ and d=$\log(\frac{1}{\delta})$. The depth d denotes the number of pairwise-independent hash functions employed by the algorithm and there exists a one-to-one correspondence between the rows and the set of hash functions. Each of these hash functions h_k:$\{\text{x}_1 \ldots \text{x}_N\} \rightarrow \{1 \ldots w\}, 1 \leq k \leq d$, takes a word pair from the input stream and maps it into a counter indexed by the corresponding hash function. For example, $h_2(\text{x}) = 10$ indicates that the word pair "x" is mapped to the 10^{th} position in the second row of the sketch array.

Initialize the entire sketch array with zeros.

Update Procedure: When a new word pair "x" with count c arrives, one counter in each row (as decided by its corresponding hash function) is updated by c.

$$\text{sketch}[k, h_k(\text{x})] \leftarrow \text{sketch}[k, h_k(\text{x})] + c, \;\; \forall 1 \leq k \leq d$$

Query Procedure: Since multiple word pairs can get hashed to the same position, the frequency stored by each position is guaranteed to overestimate the true count. Thus, to answer the point query for a given word pair, we return minimum over all the positions indexed by the k hash functions. The answer to Query(x): $\hat{c} = \min_k \; \text{sketch}[k, h_k(\text{x})]$

Both update and query procedures involve evaluating d hash functions and reading of all the values in those indices and hence both these procedures are linear in the number of hash functions. Hence both these steps require $O(\log(\frac{1}{\delta}))$ time. In our experiments (see Section 3.1), we found that a small number of hash functions are sufficient and we use d=5. Hence, the update and query operations take only a constant time. The space used by the algorithm is the size of the array i.e. wd counters, where w is the width of each row.

2.1.2 Properties

Apart from the advantages of being space efficient, and having constant update and constant querying time, the Count-Min sketch has also other advantages that makes it an attractive choice for NLP applications.

- Linearity: Given two sketches s_1 and s_2 computed (using the same parameters w and d) over different input streams, the sketch of the combined data stream can be easily obtained by adding the individual sketches in O($\frac{1}{\epsilon} \log \frac{1}{\delta}$) time which is independent of the stream size.

- The linearity is especially attractive because it allows the individual sketches to be computed independent of each other, which means that it is easy to implement it in distributed setting, where each machine computes the sketch over a sub set of corpus.

2.2 Conservative Update

Estan and Varghese introduced the idea of conservative update (Estan and Varghese, 2002) in the context of computer networking. This can easily be used with CM sketch to further improve the estimate of a point query. To update a word pair "x" with frequency c, we first compute the frequency $\hat{c}$ of this word pair from the existing data structure and the counts are updated according to:

$$\hat{c} = min_k \; \text{sketch}[k, h_k(\text{x})], \;\; \forall 1 \leq k \leq d$$

$$\text{sketch}[k, h_k(\text{x})] \leftarrow \max\{\text{sketch}[k, h_k(\text{x})], \hat{c} + c\}$$

The intuition is that, since the point query returns the minimum of all the d values, we will update a counter only if it is necessary as indicated by the above equation. Though this is a heuristic, it avoids the unnecessary updates of counter values and thus reduces the error.

In our experiments, we found that employing the conservative update reduces the Average Relative

Error (ARE) of these counts approximately by a factor of 1.5. (see Section 3.1). But unfortunately, this update can only be maintained over individual sketches in distributed setting.

3 Intrinsic Evaluations

To show the effectiveness of the CM sketch and CM sketch with conservative update (CU) in the context of NLP, we perform intrinsic evaluations. First, the intrinsic evaluations are designed to measure the error in the approximate counts returned by CM sketch compared to their true counts. Second, we compare the word pairs association rankings obtained using PMI and LLR with sketch and exact counts.

It is memory and time intensive to perform many intrinsic evaluations on large data (Ravichandran et al., 2005; Brants et al., 2007; Goyal et al., 2009). Hence, we use a subset of corpus of 2 million sentences (Subset) from Gigaword (Graff, 2003) for it. We generate words and word pairs over a window of size 7. We store exact counts of words (except stop words) in a hash table and store approximate counts of word pairs (except word pairs involving stop words) in the sketch.

3.1 Evaluating approximate sketch counts

To evaluate the amount of over-estimation error (see Section 2.1) in CM and CU counts compared to the true counts, we first group all word pairs with the same true frequency into a single bucket. We then compute the average relative error in each of these buckets. Since low-frequency word pairs are more prone to errors, making this distinction based on frequency lets us understand the regions in which the algorithm is over-estimating. Moreover, to focus on errors on low frequency counts, we have only plotted word pairs with count at most 100. Average Relative error (ARE) is defined as the average of absolute difference between the predicted and the exact value divided by the exact value over all the word pairs in each bucket.

$$\text{ARE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\text{Exact}_i - \text{Predicted}_i|}{\text{Exact}_i}$$

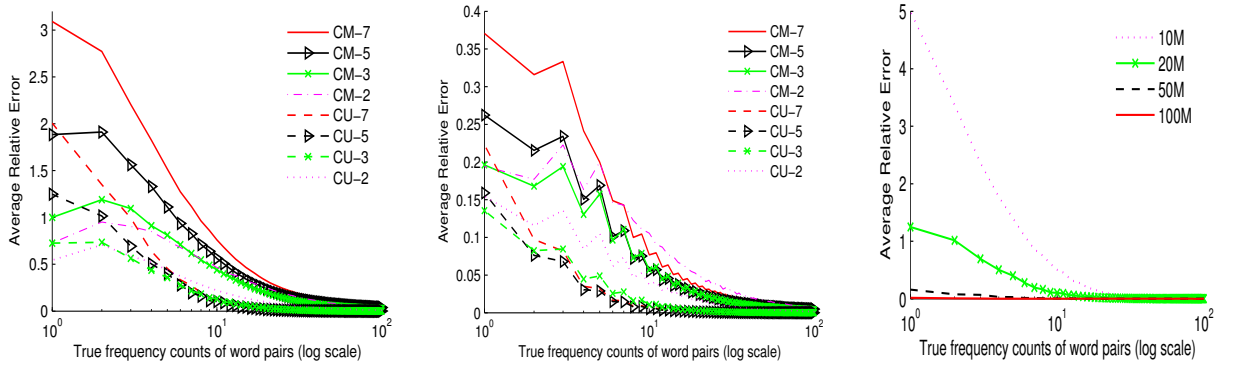
Where Exact and Predicted denotes values of exact and CM/CU counts respectively; N denotes the number of word pairs with same counts in a bucket.

In Fig. 1(a), we fixed the number of counters to 20 million ($20M$) with four bytes of memory per each counter (thus it only requires 80 MB of main memory). Keeping the total number of counters fixed, we try different values of depth (2, 3, 5 and 7) of the sketch array and in each case the width is set to $\frac{20M}{d}$. The ARE curves in each case are shown in Fig. 1(a). We can make three main observations from Figure 1(a): First it shows that most of the errors occur on low frequency word pairs. For frequent word pairs, in almost all the different runs the ARE is close to zero. Secondly, it shows that ARE is significantly lower (by a factor of 1.5) for the runs which use conservative update (CUx run) compared to the runs that use direct CM sketch (CMx run). The encouraging observation is that, this holds true for almost all different (width,depth) settings. Thirdly, in our experiments, it shows that using depth of 3 gets comparatively less ARE compared to other settings.

To be more certain about this behavior with respect to different settings of width and depth, we tried another setting by increasing the number of counters to 50 million. The curves in 1(b) follow a pattern which is similar to the previous setting. Low frequency word pairs are more prone to error compared to the frequent ones and employing conservative update reduces the ARE by a factor of 1.5. In this setting, depth 5 does slightly better than depth 3 and gets lowest ARE.

We use CU counts and depth of 5 for the rest of the paper. As 3 and 5 have lowest ARE in different settings and using 5 hash functions, we get $\delta = 0.01$ ($d = \log(\frac{1}{\delta})$ refer Section 2.1) that is probability of failure is 1 in 100, making the algorithm more robust to false positives compared with 3 hash functions, $\delta = 0.1$ with probability of failure 1 in 10.

Fig. 1(c) studies the effect of the number of counters in the sketch (the size of the two-dimensional sketch array) on the ARE with fixed depth 5. As expected, using more number of counters decreases the ARE in the counts. This is intuitive because, as the length of each row in the sketch increases, the probability of collision decreases and hence the array is more likely to contain true counts. By using 100 million counters, which is comparable to the length of the stream 88 million, we are able to achieve almost zero ARE over all the counts including the rare



(a) 20M counters (b) 50M counters (c) Different size models with depth 5

Figure 1: Compare 20 and 50 million counter models with different (width,depth) settings. The notation CMx represents the Count Min sketch with a depth of 'x' and CUx represents the CM sketch along with conservative update and depth 'x'.

ones[3]. Note that the space we *save* by not storing the exact counts is almost four times the memory that we use here because on an average each word pair is twelve characters long and requires twelve bytes (thrice the size of an integer) and 4 bytes for storing the integer count. Note, we get even bigger space savings if we work with longer phrases (phrase clustering), phrase pairs (paraphrasing/translation), and varying length n-grams (Information Extraction).

3.2 Evaluating word pairs association ranking

In this experiment, we compare the word pairs association rankings obtained using PMI and LLR with CU and exact word pair counts. We use two kinds of measures, namely recall and Spearman's correlation to measure the overlap in the rankings obtained by exact and CU counts. Intuitively, recall captures the number of word pairs that are found in both the sets and then Spearman's correlation captures if the relative order of these common word pairs is preserved in both the rankings. In our experimental setup, if the rankings match exactly, then we get a recall (R) of 100% and a correlation (ρ) of 1.

The results with respect to different sized counter (20 million ($20M$), 50 million ($50M$)) models are shown in Table 1. If we compare the second and third column of the table using PMI and LLR for $20M$ counters, we get exact rankings for LLR compared to PMI while comparing Top$K$ word pairs. The explanation for such a behavior is: since we are not throwing away any infrequent word pairs, PMI will rank pairs with low frequency counts higher (Church and Hanks, 1989). Hence, we are evaluating the PMI values for rare word pairs and we need counters linear in size of stream to get almost perfect ranking. This is also evident from the fourth column for $50M$ of the Table 1, where CU PMI ranking gets close to the optimal as the number of counters approaches stream size.

# Cs	20M				50M			
AS	PMI		LLR		PMI		LLR	
TopK	*R*	ρ	*R*	ρ	*R*	ρ	*R*	ρ
50	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
100	.98	.94	1.0	1.0	1.0	1.0	1.0	1.0
500	.80	.98	1.0	1.0	.98	1.0	1.0	1.0
1000	.56	.99	1.0	1.0	.96	.99	1.0	1.0
5000	.35	.90	1.0	1.0	.85	.99	1.0	1.0
10000	.38	.55	1.0	1.0	.81	.95	1.0	1.0

Table 1: Evaluating the PMI and LLR rankings obtained using CM sketch with conservative update (CU) and Exact counts

However, in some NLP problems, we are not interested in low-frequency items. In such cases, even using space less than linear in number of counters would suffice. In our extrinsic evaluations, we show that using space less than the length of the stream does not degrade the performance.

4 Extrinsic Evaluations

4.1 Data

Gigaword corpus (Graff, 2003) and a 50% portion of a copy of web crawled by (Ravichandran et al.,

[3]Even with other datasets we found that using counters linear in the size of the stream leads to ARE close to zero $\forall$ counts.

2005) are used to compute counts of words and word pairs. For both the corpora, we split the text into sentences, tokenize and convert into lower-case. We generate words and word pairs over a window of size 7. We use four different sized corpora: SubSet (used for intrinsic evaluations in Section 3), Gigaword (GW), GigaWord + 20% of web data (GWB20), and GigaWord + 50% of web data (GWB50). Corpus Statistics are shown below. We store exact counts of words in a hash table and store approximate counts of word pairs in the sketch. Hence, the stream size in our case is the total number of word pairs in a corpus.

Corpus	Subset	GW	GWB20	GWB50
Unzipped Size (GB)	.32	9.8	22.8	49
# of sentences (Million)	2.00	56.78	191.28	462.60
Stream Size (Billion)	.088	2.67	6.05	13.20

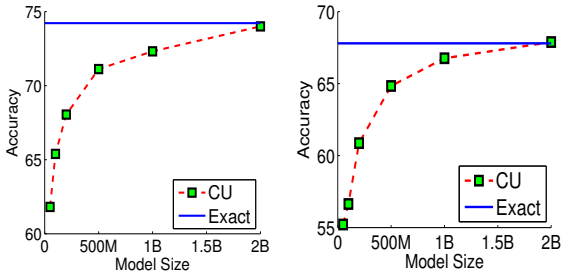
4.2 Semantic Orientation

Given a word, the task of finding the Semantic Orientation (SO) (Turney and Littman, 2003) of the word is to identify if the word is more likely to be used in positive or negative sense. We use a similar framework as used by the authors to infer the SO. We take the seven positive words (good, nice, excellent, positive, fortunate, correct, and superior) and the seven negative words (bad, nasty, poor, negative, unfortunate, wrong, and inferior) used in (Turney and Littman, 2003) work. The SO of a given word is calculated based on the strength of its association with the seven positive words, and the strength of its association with the seven negative words. We compute the SO of a word "w" as follows:

$$\text{SO-AS(w)} = \sum_{p \in Pwords} AS(p,w) - \sum_{n \in Nwords} AS(n,w)$$

Where, Pwords and Nwords denote the seven positive and negative prototype words respectively. We use PMI and LLR to compute association scores (AS). If this score is positive, we predict the word as positive. Otherwise, we predict it as negative.

We use the General Inquirer lexicon[4] (Stone et al., 1966) as a benchmark to evaluate the semantic orientation scores similar to (Turney and Littman, 2003) work. Words with multiple senses have multiple entries in the lexicon, we merge these entries for our experiment. Our test set consists of 1597 positive and 1980 negative words. Accuracy is used as an evaluation metric and is defined as the percentage of number of correctly identified SO words.



(a) SO PMI (b) SO LLR

Figure 2: Evaluating Semantic Orientation using PMI and LLR with different number of counters of CU sketch built using Gigaword.

4.2.1 Varying sketch size

We evaluate SO of words using PMI and LLR on Gigaword (9.8GB). We compare approximate SO computed using varying sizes of CU sketches: 50 million ($50M$), $100M$, $200M$, $500M$, 1 billion ($1B$) and 2 billion ($2B$) counters with Exact SO. To compute these scores, we count the number of individual words w_1 and w_2 and the pair (w_1,w_2) within a window of size 7. Note that computing the exact counts of all word pairs on these corpora is computationally expensive and memory intensive, so we consider only those pairs in which one word appears in the prototype list and the other word appears in the test set.

First, if we look at the Exact SO using PMI and LLR in Figure 2(a) and 2(b) respectively, it shows that using PMI, we get about 6 points higher accuracy than LLR on this task (The 95% statistical significance boundary for accuracy is about $\pm$ 1.5.). Second, for both PMI and LLR, having more number of counters improve performance.[5] Using $2B$ counters, we get the same accuracy as Exact.

[4]The General Inquirer lexicon is freely available at `http://www.wjh.harvard.edu/~inquirer/`

[5]We use maximum of $2B$ counters (8GB main memory), as most of the current desktop machines have at most 8GB RAM.

4.2.2 Effect of Increasing Corpus Size

We evaluate SO of words on three different sized corpora (see Section 4.1): GW (9.8GB), GWB20 (22.8GB), and GWB50 (49GB). First, since for this task using PMI performs better than LLR, so we will use PMI for this experiment. Second, we will fix number of counters to $2B$ (CU-2B) as it performs the best in Section 4.2.1. Third, we will compare the CU-2B counter model with the Exact over increasing corpus size.

We can make several observations from the Figure 3: • It shows that increasing the amount of data improves the accuracy of identifying the SO of a word. We get an absolute increase of 5.5 points in accuracy, when we add 20% Web data to GigaWord (GW). Adding 30% more Web data (GWB50), gives a small increase of 1.3 points in accuracy which is not even statistically significant. • Second, CU-2B performs as good as exact for all corpus sizes. • Third, the number of $2B$ counters (bounded space) is less than the length of stream for GWB20 ($6.05B$), and GWB50 ($13.2B$). Hence, it shows that using counters less than the stream length does not degrade the performance. • These results are also comparable to Turney's (2003) state-of-the-art work where they report an accuracy of 82.84%. Note, they use a billion word corpus which is larger than GWB50.

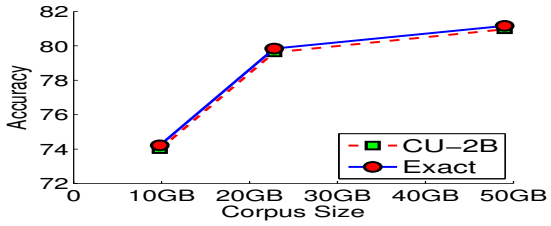


Figure 3: Evaluating Semantic Orientation of words with Exact and CU counts with increase in corpus size

4.3 Distributional Similarity

Distributional similarity is based on the distributional hypothesis that similar terms appear in similar contexts (Firth, 1968; Harris, 1954). The context vector for each term is represented by the strength of association between the term and each of the lexical, semantic, syntactic, and/or dependency units that co-occur with it[6]. We use PMI and LLR to compute association score (AS) between the term and each of the context to generate the context vector. Once, we have context vectors for each of the terms, cosine similarity measure returns distributional similarity between terms.

4.3.1 Efficient Distributional Similarity

We propose an efficient approach for computing distributional similarity between word pairs using CU sketch. In the first step, we traverse the corpus and store counts of all words (except stop words) in hash table and all word pairs (except word pairs involving stop words) in sketch. In the second step, for a target word "x", we consider all words (except infrequent contexts which appear less than or equal to 10.) as plausible context (since it is faster than traversing the whole corpus.), and query the sketch for vocabulary number of word pairs, and compute approximate AS between word-context pairs. We maintain only top K AS scores[7] contexts using priority queue for every target word "x" and save them onto the disk. In the third step, we use cosine similarity using these approximate top K context vectors to compute efficient distributional similarity.

The efficient distributional similarity using sketches has following advantages:

- It can return semantic similarity between any word pairs that are stored in the sketch.
- It can return the similarity between word pairs in time O(K).
- We do not store word pairs explicitly, and use fixed number of counters, hence the overall space required is bounded.
- The additive property of sketch (Sec. 2.1.2) enables us to parallelize most of the steps in the algorithm. Thus it can be easily extended to very large amounts of text data.

We use two test sets which consist of word pairs, and their corresponding human rankings. We generate the word pair rankings using efficient distributional similarity. We report the spearman's rank

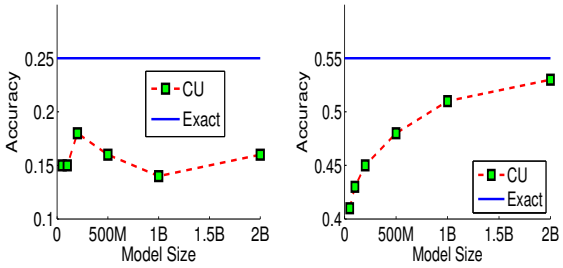
[6]Here, the context for a target word "x" is defined as words appear within a window of size 7.

[7]For this work, we use $K = 1000$ which is not tuned.

correlation[8] coefficient (ρ) between the human and distributional similarity rankings. The two test sets are:

1. **WS-353** (Finkelstein et al., 2002) is a set of 353 word pairs.

2. **RG-65**: (Rubenstein and Goodenough, 1965) is set of 65 word pairs.



(a) Word Similarity PMI (b) Word Similarity LLR

Figure 4: Evaluating Distributional Similarity between word pairs on WS-353 test set using PMI and LLR with different number of counters of CU sketch built using Gigaword data-set.

4.3.2 Varying sketch size

We evaluate efficient distributional similarity between between word pairs on WS-353 test set using PMI and LLR association scores on Gigaword (9.8GB). We compare different sizes of CU sketch (similar to SO evaluation): 50 million ($50M$), $100M$, $200M$, $500M$, 1 billion ($1B$) and 2 billion ($2B$) counters with the Exact word pair counts. Here again, computing the exact counts of all word-context pairs on these corpora is time, and memory intensive, we generate context vectors for only those words which are present in the test set.

First, if we look at word pair ranking using exact PMI and LLR across Figures 4(a) and 4(b) respectively, it shows that using LLR, we get better ρ of .55 compared to ρ of .25 using PMI on this task (The 95% statistical significance boundary on ρ for WS-353 is about $\pm$ .08). The explanation for such a behavior is: PMI rank context pairs with low frequency counts higher (Church and Hanks, 1989) compared to frequent ones which are favored by LLR. Second, for PMI in Fig. 4(a), having more counters does not improve ρ. Third, for LLR in Fig. 4(b), having more number of counters improve performance and using $2B$ counters, we get ρ close to the Exact.

[8]To calculate the Spearman correlations values are transformed into ranks (if tied ranks exist, average of ranks is taken), and we calculate the Pearson correlation on them.

Test Set	**WS-353**			**RG-65**		
Model	GW	$GWB20$	$GWB50$	GW	$GWB20$	$GWB50$
Agirre			.64			.75
Exact	.55	.55	.62	.65	.72	.74
CU-2B	.53	.58	.62	.66	.72	.74

Table 2: Evaluating word pairs ranking with Exact and CU counts. Scores are evaluated using ρ metric.

4.3.3 Effect of Increasing Corpus Size

We evaluate efficient distributional similarity between word pairs using three different sized corpora: GW (9.8GB), GWB20 (22.8GB), and GWB50 (49GB) on two test sets: WS-353, and RG-65. First, since for this task using LLR performs better than PMI, so we will use LLR for this experiment. Second, we will fix number of counters to $2B$ (CU-2B) as it performs the best in Section 4.2.1. Third, we will compare the CU-2B counter model with the Exact over increasing corpus size. We also compare our results against the state-of-the-art results (Agirre) for distributional similarity (Agirre et al., 2009). We report their results of context window of size 7.

We can make several observations from the Table 2: • It shows that increasing the amount of data is not substantially improving the accuracy of word pair rankings over both the test sets. • Here again, CU-2B performs as good as exact for all corpus sizes. • CU-2B and Exact performs same as the state-of-the-art system. • The number of $2B$ counters (bounded space) is less than the length of stream for GWB20 ($6.05B$), and GWB50 ($13.2B$). Hence, here again it shows that using counters less than the stream length does not degrade the performance.

5 Dependency Parsing

Recently, maximum spanning tree (MST) algorithms for dependency parsing (McDonald et al., 2005) have shown great promise, primarily in supervised settings. In the MST framework, words in a sentence form nodes in a graph, and connections between nodes indicate how "related" they are. A maximum spanning tree algorithm constructs a de-

pendency parse by linking together "most similar" words. Typically the weights on edges in the graph are parameterized as a linear function of features, with weight learned by some supervised learning algorithm. In this section, we ask the question: can word association scores be used to derive syntactic structures in an *unsupervised* manner?

A first pass answer is: clearly not. Metrics like PMI would assign high association scores to rare word pairs (mostly content words) leading to incorrect parses. Metrics like LLR would assign high association scores to frequent words, also leading to incorrect parses. However, with a *small amount* of linguistic side information (Druck et al., 2009; Naseem et al., 2010), we see that these issues can be overcome. In particular, we see that large data + a little linguistics > fancy unsupervised learning algorithms.

5.1 Graph Definition

Our approach is conceptually simple. We construct a graph over nodes in the sentence with a unique "root" node. The graph is directed and fully connected, and for any two words in positions i and j, the *weight* from word i to word j is defined as:

$$w_{ij} = \alpha^{\text{asc}}\text{asc}(w_i, w_j) - \alpha^{\text{dist}}\text{dist}(i-j) + \alpha^{\text{ling}}\text{ling}(t_i, t_j)$$

Here, $\text{asc}(w_i, w_j)$ is a association score such as PMI or LLR computed using approximate counts from the sketch. Similarly, $\text{dist}(i-j)$ is a simple parameterized model of distances that favors short dependencies. We use a simple unnormalized (log) Laplacian prior of the form $\text{dist}(i-j) = -|i-j-1|$, centered around 1 (encouraging short links to the right). It is negated because we need to convert distances to similarities.

The final term, $\text{ling}(t_i, t_j)$ asks: according to some simple linguistic knowledge, how likely is if that the (gold standard) part of speech tag associated with word i points at that associated with word j? For this, we use the same linguistic information used by (Naseem et al., 2010), which does *not* encode direction information. These rules are: `root` → { `aux, verb` }; `verb` → { `noun, pronoun, adverb, verb` }; `aux` → { `verb` }; `noun` → { `adj, art, noun, num` }; `prep` → { `noun` }; `adj` → { `adv` }. We simply give an additional weight of 1 to any edge that agrees with one of these linguistic rules.

	len ≤ 10	**len ≤ 20**	**all**
COHEN-DIRICHLET	45.9	39.4	34.9
COHEN-BEST	59.4	45.9	40.5
ORACLE	75.1	66.6	63.0
BASELINE+LING	42.4	33.8	29.7
BASELINE	33.5	30.4	28.9
CU-2B LLR OPTIMAL	62.4 ± *7.7*	51.1 ± *3.2*	41.1 ± *1.9*
CU-2B PMI OPTIMAL	63.3 ± *7.8*	52.0 ± *3.2*	41.1 ± *2.0*
CU-2B LLR BALANCED	49.1 ± *7.6*	43.6 ± *3.3*	37.2 ± *1.9*
CU-2B PMI BALANCED	**49.5** ± ***8.0***	**45.0** ± ***3.2***	**38.3** ± ***2.0***
CU-2B LLR SEMISUP	55.7 ± *0.0*	44.1 ± *0.0*	39.4 ± *0.0*
CU-2B PMI SEMISUP	**56.5** ± ***0.0***	**45.8** ± ***0.0***	**39.9** ± ***0.0***

Table 3: Comparing CU-2B build on GWB50 + a little linguistics v/s fancy unsupervised learning algorithms.

5.2 Parameter Setting

The remaining issue is setting the interpolation parameters α associated with each of these scores. This is a difficult problem in purely unsupervised learning. We report results on three settings. First, the OPTIMAL setting is based on grid search for optimal parameters. This is an oracle result based on grid search over two of the three parameters (holding the third fixed at 1). In our second approach, BALANCED, we normalize the three components to "compete" equally. In particular, we scale and translate all three components to have zero mean and unit variance, and set the αs to all be equal to one. Finally, our third approach, SEMISUP, is based on using a small amount of labeled data to set the parameters. In particular, we use 10 labeled sentences to select parameters based on the same grid search as the OPTIMAL setting. Since this relies heavily on which 10 sentences are used, we repeat this experiment 20 times and report averages.

5.3 Experiments

Our experiments are on a dependency-converted version of section 23 of the Penn Treebank using modified Collins' head finding rules. We measure accuracies as *directed*, unlabeled dependency accuracy. We separately report results of sentences of length at most 10, at most 20 and finally of all length. Note that there is no training or cross-validation: we simply run our MST parser on test data directly.

The results of the parsing experiments are shown

in Table 3. We compare against the following alternative systems. The first, Cohen-Dirichlet and Cohen-Best, are previously reported state-of-the-art results for unsupervised Bayesian dependency parsing (Cohen and Smith, 2010). The first is results using a simple Dirichlet prior; the second is the best reported results for any system from that paper.

Next, we compare against an "oracle" system that uses LLR extracted from the training data for the Penn Treebank, where the LLR is based on the probability of observing an edge given two words. This is not a true oracle in the sense that we *might* be able to do better, but it is unlikely. The next two baseline system are simple right branching baseline trees. The Baseline system is a purely right-branching tree. The Baseline+Ling system is one that is right branching *except* that it can only create edges that are compatible with the linguistic rules, provided a relevant rule exists. For short sentences, this is competitive with the Dirichlet prior results.

Finally we report variants of our approach using association scores computed on the GWB50 using CU sketch with 2 billion counters. We experiment with two association scores: LLR and PMI. For each measure, we report results based on the three approaches described earlier for setting the α hyperparameters. Error bars for our approaches are 95% confidence intervals based on bootstrap resampling.

The results show that, for this task, PMI seems slightly better than LLR, across the board. The OPTIMAL performance (based on tuning two hyperparameters) is amazingly strong: clearly beating out all the baselines, and only about 15 points behind the ORACLE system. Using the BALANCED approach causes a degradation of only 3 points from the OPTIMAL on sentences of all lengths. In general, the balancing approach seems to be slightly worse than the semi-supervised approach, except on very short sentences: for those, it is substantially better. Overall, though, the results for both Balanced and Semisup are competitive with state-of-the-art unsupervised learning algorithms.

6 Discussion and Conclusion

The advantage of using sketch in addition to being memory and time efficient is that it contains counts for all word pairs and hence can be used to compute association scores like PMI and LLR between any word pairs. We show that using sketch counts in our experiments, on the three tasks, we get performance comparable to Exact word pair counts setting and state-of-the-art system. Our method scales to 49 GB of unzipped web data using bounded space of 2 billion counters (8 GB memory). Moreover, the linearity property of the sketch makes it scalable and usable in distributed setting. Association scores and counts from sketch can be used for more NLP tasks like small-space randomized language models, word sense disambiguation, spelling correction, relation learning, paraphrasing, and machine translation.

Acknowledgments

The authors gratefully acknowledge the support of NSF grant IIS-0712764 and Google Research Grant for Large-Data NLP. Thanks to Suresh Venkatasubramanian and Jagadeesh Jagarlamudi for useful discussions and the anonymous reviewers for many helpful comments.

References

Charu C. Aggarwal and Philip S. Yu. 2010. On classification of high-cardinality data streams. In *SDM'10*, pages 802–813.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL '09: Proceedings of HLT-NAACL*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-CoNLL*.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312:3–15, January.

K. Church and P. Hanks. 1989. Word Association Norms, Mutual Information and Lexicography. In *Proceedings of ACL*, pages 76–83, Vancouver, Canada, June.

S. B. Cohen and N. A. Smith. 2010. Covariance in unsupervised learning of probabilistic grammars. *Journal of Machine Learning Research*, 11:3017–3051.

Graham Cormode and Marios Hadjieleftheriou. 2008. Finding frequent items in data streams. In *VLDB*.

Graham Cormode and S. Muthukrishnan. 2004. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2009. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 360–368, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. 2010. Pan-private streaming algorithms. In *In Proceedings of ICS*.

Cristian Estan and George Varghese. 2002. New directions in traffic measurement and accounting. *SIGCOMM Comput. Commun. Rev.*, 32(4).

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2002. Placing search in context: The concept revisited. In *ACM Transactions on Information Systems*.

J. Firth. 1968. A synopsis of linguistic theory 1930-1955. In F. Palmer, editor, *Selected Papers of J. R. Firth*. Longman.

Amit Goyal, Hal Daumé III, and Suresh Venkatasubramanian. 2009. Streaming for large scale NLP: Language modeling. In *NAACL*.

Amit Goyal, Jagadeesh Jagarlamudi, Hal Daumé III, and Suresh Venkatasubramanian. 2010a. Sketch techniques for scaling distributional similarity to the web. In *GEMS workshop at ACL*, Uppsala, Sweden.

Amit Goyal, Jagadeesh Jagarlamudi, Hal Daumé III, and Suresh Venkatasubramanian. 2010b. Sketching techniques for Large Scale NLP. In *6th WAC Workshop at NAACL-HLT*.

D. Graff. 2003. English Gigaword. Linguistic Data Consortium, Philadelphia, PA, January.

Z. Harris. 1954. Distributional structure. *Word 10 (23)*, pages 146–162.

Abby Levenberg and Miles Osborne. 2009. Stream-based randomised language models for SMT. In *EMNLP*, August.

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. 2010. Stream-based translation models for statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 394–402. Association for Computational Linguistics.

Ping Li, Kenneth Ward Church, and Trevor Hastie. 2008. One sketch for all: Theory and application of conditional random sampling. In *Neural Information Processing Systems*, pages 953–960.

G. S. Manku and R. Motwani. 2002. Approximate frequency counts over data streams. In *VLDB*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

S. Muthukrishnan. 2005. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2).

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1234–1244. Association for Computational Linguistics.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP*.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *Proceedings of ACL*.

H. Rubenstein and J.B. Goodenough. 1965. Contextual correlates of synonymy. *Computational Linguistics*, 8:627–633.

Florin Rusu and Alin Dobra. 2007. Statistical analysis of sketch estimators. In *SIGMOD '07*. ACM.

Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. 2010. Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. In *Proceedings of the 5th USENIX conference on Hot topics in security*, HotSec'10, pages 1–8, Berkeley, CA, USA. USENIX Association.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and S.V.N. Vishwanathan. 2009. Hash kernels for structured data. *J. Mach. Learn. Res.*, 10:2615–2637, December.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21:315–346, October.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of COLING 2008*.

Benjamin Van Durme and Ashwin Lall. 2009a. Probabilistic counting with randomized storage. In *IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence*.

Benjamin Van Durme and Ashwin Lall. 2009b. Streaming pointwise mutual information. In *Advances in Neural Information Processing Systems 22*.

Benjamin Van Durme and Ashwin Lall. 2010. Online generation of locality sensitive hash signatures. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 231–235, July.

Optimizing Semantic Coherence in Topic Models

David Mimno
Princeton University
Princeton, NJ 08540
mimno@cs.princeton.edu

Hanna M. Wallach
University of Massachusetts, Amherst
Amherst, MA 01003
wallach@cs.umass.edu

Edmund Talley **Miriam Leenders**
National Institutes of Health
Bethesda, MD 20892
{talleye,leenderm}@ninds.nih.gov

Andrew McCallum
University of Massachusetts, Amherst
Amherst, MA 01003
mccallum@cs.umass.edu

Abstract

Latent variable models have the potential to add value to large document collections by discovering interpretable, low-dimensional subspaces. In order for people to use such models, however, they must trust them. Unfortunately, typical dimensionality reduction methods for text, such as latent Dirichlet allocation, often produce low-dimensional subspaces (topics) that are obviously flawed to human domain experts. The contributions of this paper are threefold: (1) An analysis of the ways in which topics can be flawed; (2) an automated evaluation metric for identifying such topics that does not rely on human annotators or reference collections outside the training data; (3) a novel statistical topic model based on this metric that significantly improves topic quality in a large-scale document collection from the National Institutes of Health (NIH).

1 Introduction

Statistical topic models such as latent Dirichlet allocation (LDA) (Blei et al., 2003) provide a powerful framework for representing and summarizing the contents of large document collections. In our experience, however, the primary obstacle to acceptance of statistical topic models by users the outside machine learning community is the presence of poor quality topics. Topics that mix unrelated or loosely-related concepts substantially reduce users' confidence in the utility of such automated systems.

In general, users prefer models with larger numbers of topics because such models have greater resolution and are able to support finer-grained distinctions. Unfortunately, we have observed that there is a strong relationship between the size of topics and the probability of topics being nonsensical as judged by domain experts: as the number of topics increases, the smallest topics (number of word tokens assigned to each topic) are almost always poor quality. The common practice of displaying only a small number of example topics hides the fact that as many as 10% of topics may be so bad that they cannot be shown without reducing users' confidence.

The evaluation of statistical topic models has traditionally been dominated by either extrinsic methods (i.e., using the inferred topics to perform some external task such as information retrieval (Wei and Croft, 2006)) or quantitative intrinsic methods, such as computing the probability of held-out documents (Wallach et al., 2009). Recent work has focused on evaluation of topics as semantically-coherent concepts. For example, Chang et al. (2009) found that the probability of held-out documents is not always a good predictor of human judgments. Newman et al. (2010) showed that an automated evaluation metric based on word co-occurrence statistics gathered from Wikipedia could predict human evaluations of topic quality. AlSumait et al. (2009) used differences between topic-specific distributions over words and the corpus-wide distribution over words to identify overly-general "vacuous" topics. Finally, Andrzejewski et al. (2009) developed semi-supervised methods that avoid specific user-labeled semantic coherence problems.

The contributions of this paper are threefold: (1) To identify distinct classes of low-quality topics, some of which are not flagged by existing evaluation methods; (2) to introduce a new topic "coherence" score that corresponds well with human coherence judgments and makes it possible to identify

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 262–272,
Edinburgh, Scotland, UK, July 27–31, 2011.

specific semantic problems in topic models without human evaluations or external reference corpora; (3) to present an example of a new topic model that learns latent topics by directly optimizing a metric of topic coherence. With little additional computational cost beyond that of LDA, this model exhibits significant gains in average topic coherence score. Although the model does not result in a statistically-significant reduction in the number of topics marked "bad", the model consistently improves the topic coherence score of the ten lowest-scoring topics (i.e., results in bad topics that are "less bad" than those found using LDA) while retaining the ability to identify low-quality topics without human interaction.

2 Latent Dirichlet Allocation

LDA is a generative probabilistic model for documents $\mathcal{W} = \{\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \ldots, \boldsymbol{w}^{(D)}\}$. To generate a word token $w_n^{(d)}$ in document d, we draw a discrete topic assignment $z_n^{(d)}$ from a document-specific distribution over the T topics $\boldsymbol{\theta}_d$ (which is itself drawn from a Dirichlet prior with hyperparameter α), and then draw a word type for that token from the topic-specific distribution over the vocabulary $\boldsymbol{\phi}_{z_n^{(d)}}$. The inference task in topic models is generally cast as inferring the document–topic proportions $\{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_D\}$ and the topic-specific distributions $\{\boldsymbol{\phi}_1 \ldots, \boldsymbol{\phi}_T\}$.

The multinomial topic distributions are usually drawn from a shared symmetric Dirichlet prior with hyperparameter β, such that conditioned on $\{\boldsymbol{\phi}_t\}_{t=1}^T$ and the topic assignments $\{\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(D)}\}$, the word tokens are independent. In practice, however, it is common to deal directly with the "collapsed" distributions that result from integrating over the topic-specific multinomial parameters. The resulting distribution over words for a topic t is then a function of the hyperparameter β and the number of words of each type assigned to that topic, $N_{w|t}$. This distribution, known as the Dirichlet compound multinomial (DCM) or Pólya distribution (Doyle and Elkan, 2009), breaks the assumption of conditional independence between word tokens given topics, but is useful during inference because the conditional probability of a word w given topic t takes a very simple form: $P(w \mid t, \beta) = \frac{N_{w|t}+\beta}{N_t+|\mathcal{V}|\,\beta}$, where $N_t = \sum_{w'} N_{w'|t}$ and $|\mathcal{V}|$ is the vocabulary size.

The process for generating a sequence of words from such a model is known as the simple Pólya urn model (Mahmoud, 2008), in which the initial probability of word type w in topic t is proportional to β, while the probability of each subsequent occurrence of w in topic t is proportional to the number of times w has been drawn in that topic plus β. Note that this unnormalized weight for each word type depends only on the count of that word type, and is independent of the count of any other word type w'. Thus, in the DCM/Pólya distribution, drawing word type w must *decrease* the probability of seeing all other word types $w' \neq w$. In a later section, we will introduce a topic model that substitutes a *generalized* Pólya urn model for the DCM/Pólya distribution, allowing a draw of word type w to *increase* the probability of seeing certain other word types.

For real-world data, documents $\mathcal{W}$ are observed, while the corresponding topic assignments $\mathcal{Z}$ are unobserved and may be inferred using either variational methods (Blei et al., 2003; Teh et al., 2006) or MCMC methods (Griffiths and Steyvers, 2004). Here, we use MCMC methods—specifically Gibbs sampling (Geman and Geman, 1984), which involves sequentially resampling each topic assignment $z_n^{(d)}$ from its conditional posterior given the documents $\mathcal{W}$, the hyperparameters α and β, and $\mathcal{Z}_{\setminus d,n}$ (the current topic assignments for all tokens other than the token at position n in document d).

3 Expert Opinions of Topic Quality

Concentrating on 300,000 grant and related journal paper abstracts from the National Institutes of Health (NIH), we worked with two experts from the National Institute of Neurological Disorders and Stroke (NINDS) to collaboratively design an expert-driven topic annotation study. The goal of this study was to develop an annotated set of baseline topics, along with their salient characteristics, as a first step towards automatically identifying and inferring the kinds of topics desired by domain experts.[1]

3.1 Expert-Driven Annotation Protocol

In order to ensure that the topics selected for annotation were within the NINDS experts' area of expertise, they selected 148 topics (out of 500), all associated with areas funded by NINDS. Each topic

[1] All evaluated models will be released publicly.

t was presented to the experts as a list of the thirty most probable words for that topic, in descending order of their topic-specific "collapsed" probabilities, $\frac{N_{w|t}+\beta}{N_t+|\mathcal{V}|\,\beta}$. In addition to the most probable words, the experts were also given metadata for each topic: The most common sequences of two or more consecutive words assigned to that topic, the four topics that most often co-occurred with that topic, the most common IDF-weighted words from titles of grants, thesaurus terms, NIH institutes, journal titles, and finally a list of the highest probability grants and PubMed papers for that topic.

The experts first categorized each topic as one of three types: "research", "grant mechanisms and publication types" or "general".[2] The quality of each topic ("good", "intermediate", or "bad") was then evaluated using criteria specific to the type of topic. In general, topics were only annotated as "good" if they contained words that could be grouped together as a single coherent concept. Additionally, each "research" topic was only considered to be "good" if, in addition to representing a single coherent concept, the aggregate content of the set of documents with appreciable allocations to that topic clearly contained text referring to the concept inferred from the topic words. Finally, for each topic marked as being either "intermediate" or "bad", one or more of the following problems (defined by the domain experts) was identified, as appropriate:

- **Chained:** every word is connected to every other word through some pairwise word chain, but not all word pairs make sense. For example, a topic whose top three words are "acids", "fatty" and "nucleic" consists of two distinct concepts (i.e., acids produced when fats are broken down versus the building blocks of DNA and RNA) chained via the word "acids".
- **Intruded:** either two or more unrelated sets of related words, joined arbitrarily, or an otherwise good topic with a few "intruder" words.
- **Random:** no clear, sensical connections between more than a few pairs of words.
- **Unbalanced:** the top words are all logically connected to each other, but the topic combines very general and specific terms (e.g., "signal transduction" versus "notch signaling").

Examples of a good general topic, a good research topic, and a chained research topic are in Table 1.

3.2 Annotation Results

The experts annotated the topics independently and then aggregated their results. Interestingly, no topics were ever considered "good" by one expert and "bad" by the other—when there was disagreement between the experts, one expert always believed the topic to be "intermediate." In such cases, the experts discussed the reasons for their decisions and came to a consensus. Of the 148 topics selected for annotation, 90 were labeled as "good," 21 as "intermediate," and 37 as "bad." Of the topics labeled as "bad" or "intermediate," 23 were "chained," 21 were "intruded," 3 were "random," and 15 were "unbalanced". (The annotators were permitted to assign more than one problem to any given topic.)

4 Automated Metrics for Predicting Expert Annotations

The ultimate goal of this paper is to develop methods for building models with large numbers of specific, high-quality topics from domain-specific corpora. We therefore explore the extent to which information already contained in the documents being modeled can be used to assess topic quality.

In this section we evaluate several methods for ranking the quality of topics and compare these rankings to human annotations. No method is likely to perfectly predict human judgments, as individual annotators may disagree on particular topics. For an application involving removing low quality topics we recommend using a weighted combination of metrics, with a threshold determined by users.

4.1 Topic Size

As a simple baseline, we considered the extent to which topic "size" (as measured by the number of tokens assigned to each topic via Gibbs sampling) is a good metric for assessing topic quality. Figure 1 (top) displays the topic size (number of tokens assigned to that topic) and expert annotations ("good", "intermediate", "bad") for the 148 topics manually labeled by annotators as described above. This figure suggests that topic size is a reasonable predic-

[2] Equivalent to "vacuous topics" of AlSumait et al. (2009).

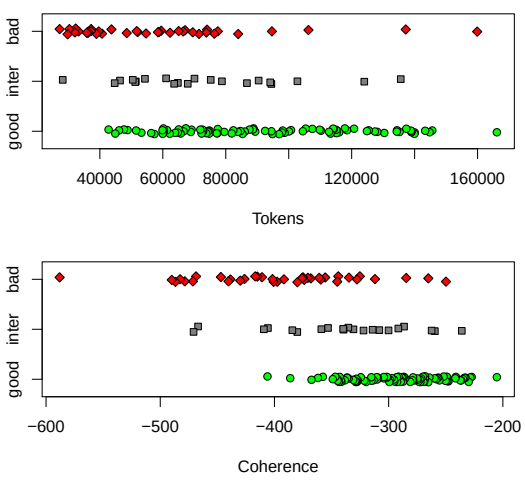


Figure 1: **Topic size is a good indicator of quality; the new *coherence* metric is better**. Top shows expert-rated topics ranked by topic size (AP 0.89, AUC 0.79), bottom shows same topics ranked by coherence (AP 0.94, AUC 0.87). Random jitter is added to the y-axis for clarity.

tor of topic quality. Although there is some overlap, "bad" topics are generally smaller than "good" topics. Unfortunately, this observation conflicts with the goal of building highly specialized, domain-specific topic models with many high-quality, fine-grained topics—in such models the majority of topics will have relatively few tokens assigned to them.

4.2 Topic Coherence

When displaying topics to users, each topic t is generally represented as a list of the $M = 5, \ldots, 20$ most probable words for that topic, in descending order of their topic-specific "collapsed" probabilities. Although there has been previous work on automated generation of labels or headings for topics (Mei et al., 2007), we choose to work only with the ordered list representation. Labels may obscure or detract from fundamental problems with topic coherence, and better labels don't make bad topics good.

The expert-driven annotation study described in section 3 suggests that three of the four types of poor-quality topics ("chained," "intruded" and "random") could be detected using a metric based on the co-occurrence of words within the documents being modeled. For "chained" and "intruded" topics, it is likely that although pairs of words belonging to a single concept will co-occur within a single document (e.g., "nucleic" and "acids" in documents about DNA), word pairs belonging to different concepts (e.g., "fatty" and "nucleic") will not. For random topics, it is likely that few words will co-occur.

This insight can be used to design a new metric for assessing topic quality. Letting $D(v)$ be the *document frequency* of word type v (i.e., the number of documents with least one token of type v) and $D(v, v')$ be *co-document frequency* of word types v and v' (i.e., the number of documents containing one or more tokens of type v and at least one token of type v'), we define *topic coherence* as

$$C(t; V^{(t)}) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D(v_m^{(t)}, v_l^{(t)}) + 1}{D(v_l^{(t)})}, \quad (1)$$

where $V^{(t)} = (v_1^{(t)}, \ldots, v_M^{(t)})$ is a list of the M most probable words in topic t. A smoothing count of 1 is included to avoid taking the logarithm of zero.

Figure 1 shows the association between the expert annotations and both topic size (top) and our coherence metric (bottom). We evaluate these results using standard ranking metrics, average precision and the area under the ROC curve. Treating "good" topics as positive and "intermediate" or "bad" topics as negative, we get average precision values of 0.89 for topic size vs. 0.94 for coherence and AUC 0.79 for topic size vs. 0.87 for coherence. We performed a logistic regression analysis on the binary variable "is this topic bad". Using topic size alone as a predictor gives AIC (a measure of model fit) 152.5. Coherence alone has AIC 113.8 (substantially better). Both predictors combined have AIC 115.8: the simpler coherence alone model provides the best performance. We tried weighting the terms in equation 1 by their corresponding topic–word probabilities and and by their position in the sorted list of the M most probable words for that topic, but we found that a uniform weighting better predicted topic quality.

Our topic coherence metric also exhibits good qualitative behavior: of the 20 best-scoring topics, 18 are labeled as "good," one is "intermediate" ("unbalanced"), and one is "bad" (combining "cortex" and "fmri", words that commonly co-occur, but are conceptually distinct). Of the 20 worst scoring topics, 15 are "bad," 4 are "intermediate," and only one (with the 19th worst coherence score) is "good."

Our coherence metric relies only upon word co-occurrence statistics gathered from the corpus being modeled, and does not depend on an external reference corpus. Ideally, all such co-occurrence information would already be accounted for in the model. We believe that one of the main contributions of our work is demonstrating that standard topic models *do not* fully utilize available co-occurrence information, and that a held-out reference corpus is therefore not required for purposes of topic evaluation.

Equation 1 is very similar to pointwise mutual information (PMI), but is more closely associated with our expert annotations than PMI (which achieves AUC 0.64 and AIC 170.51). PMI has a long history in language technology (Church and Hanks, 1990), and was recently used by Newman et al. (2010) to evaluate topic models. When expressed in terms of count variables as in equation 1, PMI includes an additional term for $D(v_m^{(t)})$. The improved performance of our metric over PMI implies that what matters is *not* the difference between the joint probability of words m and l and the product of marginals, but the *conditional* probability of each word given the each of the higher-ranked words in the topic.

In order to provide intuition for the behavior of our topic coherence metric, table 1 shows three example topics and their topic coherence scores. The first topic, related to grant-funded training programs, is one of the best-scoring topics. All pairs of words have high co-document frequencies. The second topic, on neurons, is more typical of quality "research" topics. Overall, these words occur less frequently, but generally occur moderately interchangeably: there is little structure to their covariance. The last topic is one of the lowest-scoring topics. Its co-document frequency matrix is shown in table 2. The top two words are closely related: 487 documents include "aging" at least once, 122 include "lifespan", and 55 include both. Meanwhile, the third word "globin" occurs with only one of the top seven words—the common word "human".

4.3 Comparison to word intrusion

As an additional check for both our expert annotations and our automated metric, we replicated the "word intrusion" evaluation originally introduced by Chang et al. (2009). In this task, one of the top ten most probable words in a topic is replaced with a

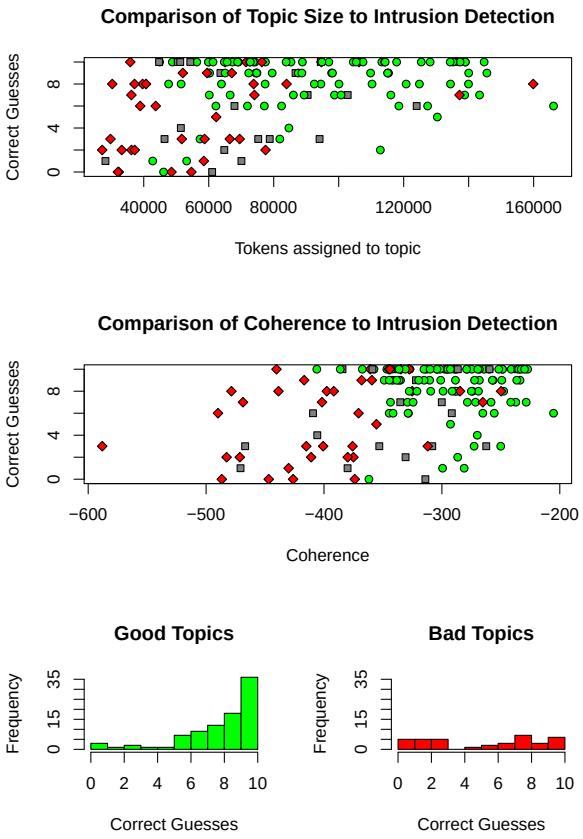


Figure 2: Top: results of the intruder selection task relative to two topic quality metrics. Bottom: marginal intruder accuracy frequencies of good and bad topics.

another word, selected at random from the corpus. The resulting set of words is presented, in a random order, to users, who are asked to identify the "intruder" word. It is very unlikely that a randomly-chosen word will be semantically related to any of the original words in the topic, so if a topic is a high quality representation of a semantically coherent concept, it should be easy for users to select the intruder word. If the topic is not coherent, there may be words in the topic that are also not semantically related to any other word, thus causing users to select "correct" words instead of the real intruder.

We recruited ten additional expert annotators from NINDS, not including our original annotators, and presented them with the intruder selection task, using the set of previously evaluated topics. Results are shown in figure 2. In the first two plots, the x-axis is one of our two automated quality met-

Table 1: Example topics (good/general, good/research, chained/research) with different coherence scores (numbers closer to zero indicate higher coherence). The chained topic combines words related to aging (indicated in plain text) and words describing blood and blood-related diseases (bold). The only connection is the common word *human*.

-167.1	students, program, summer, biomedical, training, experience, undergraduate, career, minority, student, careers, underrepresented, medical_students, week, science
-252.1	neurons, neuronal, brain, axon, neuron, guidance, nervous_system, cns, axons, neural, axonal, cortical, survival, disorders, motor
-357.2	aging, lifespan, **globin**, age_related, longevity, human, age, **erythroid**, **sickle_cell**, **beta_globin**, **hb**, senescence, adult, older, lcr

Table 2: Co-document frequency matrix for the top words in a low-quality topic (according to our coherence metric), shaded to highlight zeros. The diagonal (light gray) shows the overall document frequency for each word w. The column on the right is $N_{w|t}$. Note that "globin" and "erythroid" do not co-occur with any of the aging-related words.

aging	487	53	0	65	42	0	51	0	138	0	914
lifespan	53	122	0	15	28	0	15	0	44	0	205
globin	0	0	39	0	0	19	0	15	27	3	200
age_related	65	15	0	119	12	0	25	0	37	0	160
longevity	42	28	0	12	73	0	6	0	20	1	159
erythroid	0	0	19	0	0	69	0	8	23	1	110
age	51	15	0	25	6	0	245	1	82	0	103
sickle_cell	0	0	15	0	0	8	1	43	16	2	93
human	138	44	27	37	20	23	82	16	4347	157	91
hb	0	0	3	0	1	1	0	2	5	15	73

rics (topic size and coherence) and the y-axis is the number of annotators that correctly identified the true intruder word (accuracy). The histograms below these plots show the number of topics with each level of annotator accuracy for good and bad topics. For good topics (green circles), the annotators were generally able to detect the intruder word with high accuracy. Bad topics (red diamonds) had more uniform accuracies. These results suggest that topics with low intruder detection accuracy tend to be bad, but some bad topics can have a high accuracy. For example, spotting an intruder word in a chained topic can be easy. The low-quality topic *receptors, cannabinoid, cannabinoids, ligands, cannabis, endocannabinoid, cxcr4, [virus], receptor, sdf1*, is a typical "chained" topic, with *CXCR4* linked to *cannabinoids* only through *receptors*, and otherwise unrelated. Eight out of ten annotators correctly identified "virus" as the correct intruder. Repeating the logistic regression experiment using intruder detection accuracy as input, the AIC value is 163.18—much worse than either topic size or coherence.

5 Generalized Pólya Urn Models

Although the topic coherence metric defined above provides an accurate way of assessing topic quality, *preventing* poor quality topics from occurring in the first place is preferable. Our results in the previous section show that we can identify low-quality topics without making use of external supervision; the training data by itself contains sufficient information at least to reject poor combinations of words.

In this section, we describe a new topic model that incorporates the corpus-specific word co-occurrence information used in our coherence metric directly into the statistical topic modeling framework. It is important to note that simply disallowing words that never co-occur from being assigned to the same topic is not sufficient. Due to the power-law characteristics of language, most words are rare and will not co-occur with most other words regardless of their semantic similarity. It is rather the *degree* to which the most prominent words in a topic do not co-occur with the other most prominent words in that topic that is an indicator of topic incoherence. We therefore desire models that guide topics towards semantic similarity without imposing hard constraints.

As an example of such a model, we present a new topic model in which the occurrence of word type w in topic t increases not only the probability of seeing that word type again, but also increases the probability of seeing other related words (as determined by co-document frequencies for the corpus being modeled). This new topic model retains the document–topic component of standard LDA, but replaces the usual Pólya urn topic–word component with a *generalized* Pólya urn framework (Mahmoud, 2008).

A sequence of i.i.d. samples from a discrete distribution can be imagined as arising by repeatedly drawing a random ball from an urn, where the number of balls of each color is proportional to the probability of that color, replacing the selected ball after each draw. In a Pólya urn, each ball is replaced *along with another ball of the same color*. Samples from this model exhibit the "burstiness" property: the probability of drawing a ball of color w increases each time a ball of that color is drawn. This process represents the marginal distribution of a hierarchical model with a Dirichlet prior and a multinomial likelihood, and is used as the distribution over words for each topic in almost all previous topic models. In a *generalized* Pólya urn model, having drawn a ball of color w, A_{vw} additional balls of each color $v \in \{1, \ldots, W\}$ are returned to the urn. Given $\mathcal{W}$ and $\mathcal{Z}$, the conditional posterior probability of word w in topic t implied by this generalized model is

$$P(w \mid t, \mathcal{W}, \mathcal{Z}, \beta, \boldsymbol{A}) = \frac{\sum_v N_{v|t}\, A_{vw} + \beta}{N_t + |\mathcal{V}|\beta}, \quad (2)$$

where $\boldsymbol{A}$ is a $W \times W$ real-valued matrix, known as the *addition matrix* or *schema*. The simple Pólya urn model (and hence the conditional posterior probability of word w in topic t under LDA) can be recovered by setting the schema $\boldsymbol{A}$ to the identity matrix. Unlike the simple Pólya distribution, we do not know of a representation of the generalized Pólya urn distribution that can be expressed using a concise set of conditional independence assumptions. A standard graphical model with plate notation would therefore not be helpful in highlighting the differences between the two models, and is not shown.

Algorithm 1 shows pseudocode for a single Gibbs sweep over the latent variables $\mathcal{Z}$ in standard LDA. Algorithm 2 shows the modifications necessary to

for $d \in \mathcal{D}$ **do**
  for $w_n \in \boldsymbol{w}^{(d)}$ **do**
    $N_{z_i|d_i} \leftarrow N_{z_i|d_i} - 1$
    $N_{w_i|z_i} \leftarrow N_{w_i|z_i} - 1$
    `sample` $z_i \propto (N_{z|d_i} + \alpha_z) \frac{N_{w_i|z} + \beta}{\sum_{z'} (N_{w_i|z'} + \beta)}$
    $N_{z_i|d_i} \leftarrow N_{z_i|d_i} + 1$
    $N_{w_i|z_i} \leftarrow N_{w_i|z_i} + 1$
  end for
end for

Algorithm 1: One sweep of LDA Gibbs sampling.

for $d \in \mathcal{D}$ **do**
  for $w_n \in \boldsymbol{w}^{(d)}$ **do**
    $N_{z_i|d_i} \leftarrow N_{z_i|d_i} - 1$
    for all v **do**
      $N_{v|z_i} \leftarrow N_{v|z_i} - A_{vw_i}$
    end for
    `sample` $z_i \propto (N_{z|d_i} + \alpha_z) \frac{N_{w_i|z} + \beta}{\sum_{z'} (N_{w_i|z'} + \beta)}$
    $N_{z_i|d_i} \leftarrow N_{z_i|d_i} + 1$
    for all v **do**
      $N_{v|z_i} \leftarrow N_{v|z_i} + A_{vw_i}$
    end for
  end for
end for

Algorithm 2: One sweep of gen. Pólya Gibbs sampling, with differences from LDA highlighted in red.

support a generalized Pólya urn model: rather than subtracting exactly one from the count of the word given the old topic, sampling, and then adding one to the count of the word given the new topic, we subtract a column of the schema matrix from the entire count vector over words for the old topic, sample, and add the same column to the count vector for the new topic. As long as A is sparse, this operation adds only a constant factor to the computation.

Another property of the generalized Pólya urn model is that it is nonexchangeable—the joint probability of the tokens in any given topic is not invariant to permutation of those tokens. Inference of $\mathcal{Z}$ given $\mathcal{W}$ via Gibbs sampling involves repeatedly cycling through the tokens in $\mathcal{W}$ and, for each one, resampling its topic assignment conditioned on $\mathcal{W}$ and the current topic assignments for all tokens other than the token of interest. For LDA, the sampling distribution for each topic assignment is simply the product of two predictive probabilities, obtained by treating the token of interest as if it were the last. For a topic model with a generalized Pólya urn for the topic–word component, the sampling distribution is more complicated. Specifically, the topic–word component of the sampling distribution is no longer a simple predictive distribution—when sampling a new value for $z_n^{(d)}$, the implication of each possible value for subsequent tokens and their topic assignments must be considered. Unfortunately, this can be very computationally expensive, particularly for large corpora. There are several ways around this problem. The first is to use sequential Monte Carlo methods, which have been successfully applied to topic models previously (Canini et al., 2009). The second approach is to approximate the true Gibbs sampling distribution by treating each token as if it were the last, ignoring implications for subsequent tokens and their topic assignments. We find that this approximate method performs well empirically.

5.1 Setting the Schema A

Inspired by our evaluation metric, we define A as

$$\begin{aligned} \boldsymbol{A}_{vv} &\propto \lambda_v D(v) \\ \boldsymbol{A}_{vw} &\propto \lambda_v D(w, v) \end{aligned} \tag{3}$$

where each element is scaled by a row-specific weight λ_v and each column is normalized to sum to 1. Normalizing columns makes comparison to standard LDA simpler, because the relative effect of smoothing parameter $\beta = 0.01$ is equivalent. We set $\lambda_v = \log(D \,/\, D(v))$, the standard IDF weight used in information retrieval, which is larger for less frequent words. The column for word type w can be interpreted as word types with significant association with w. The IDF weighting therefore has the effect of increasing the strength of association for rare word types. We also found empirically that it is helpful to remove off-diagonal elements for the most common types, such as those that occur in more than 5% of documents (IDF < 3.0). Including nonzero off-diagonal values in $\boldsymbol{A}$ for very frequent types causes the model to disperse those types over many topics, which leads to large numbers of extremely similar topics. To measure this effect, we calculated the Jensen-Shannon divergence between all pairs of topic–word distributions in a given model. For a model using off-diagonal weights for all word

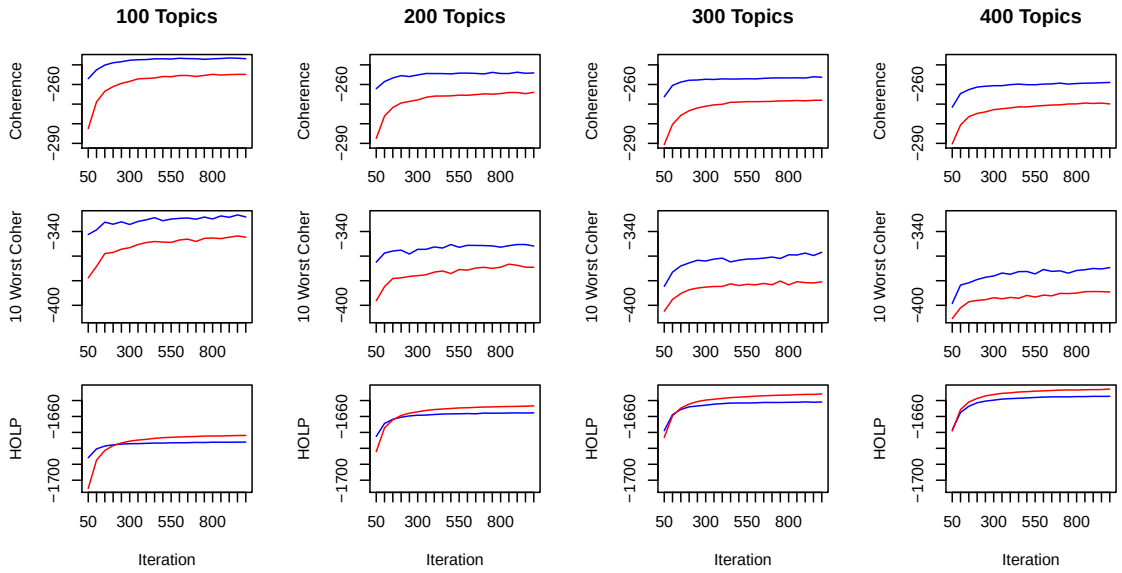


Figure 3: **Pólya urn topics (blue) have higher average coherence and converge much faster than LDA topics (red)**. The top plots show topic coherence (averaged over 15 runs) over 1000 iterations of Gibbs sampling. Error bars are not visible in this plot. The middle plot shows the average coherence of the 10 lowest scoring topics. The bottom plots show held-out log probability (in thousands) for the same models (three runs each of 5-fold cross-validation).

Name	Docs	Avg. Tok.	Tokens	Vocab
NIH	18756	114.64 ± 30.41	2150172	28702

Table 3: Data set statistics.

types, the mean of the 100 lowest divergences was $0.29 \pm .05$ (a divergence of 1.0 represents distributions with no shared support) at $T = 200$. The average divergence of the 100 most similar pairs of topics for standard LDA (i.e., $\boldsymbol{A} = \boldsymbol{I}$) is $0.67 \pm .05$. The same statistic for the generalized Pólya urn model without off-diagonal elements for word types with high document frequency is 0.822 ± 0.09.

Setting the off-diagonal elements of the schema $\boldsymbol{A}$ to zero for the most common word types also has the fortunate effect of substantially reducing preprocessing time. We find that Gibbs sampling for the generalized Pólya model takes roughly two to three times longer than for standard LDA, depending on the sparsity of the schema, due to additional bookkeeping needed before and after sampling topics.

5.2 Experimental Results

We evaluated the new model on a corpus of NIH grant abstracts. Details are given in table 3. Figure 3 shows the performance of the generalized Pólya urn model relative to LDA. Two metrics—our new topic coherence metric and the log probability of held-out documents—are shown over 1000 iterations at 50 iteration intervals. Each model was run over five folds of cross validation, each with three random initializations. For each model we calculated an overall coherence score by calculating the topic coherence for each topic individually and then averaging these values. We report the average over all 15 models in each plot. Held-out probabilities were calculated using the left-to-right method of Wallach et al. (2009), with each cross-validation fold using its own schema $\boldsymbol{A}$. The generalized Pólya model performs very well in average topic coherence, reaching levels within the first 50 iterations that match the final score. This model has an early advantage for held-out probability as well, but is eventually overtaken by LDA. This trend is consistent with Chang et al.'s observation that held-out probabilities are not always good predictors of human judgments (Chang et al., 2009). Results are consistent over $T \in \{100, 200, 300\}$.

In section 4.2, we demonstrated that our topic coherence metric correlates with expert opinions of topic quality for standard LDA. The generalized

Pólya urn model was therefore designed with the goal of directly optimizing that metric. It is possible, however, that optimizing for coherence directly could break the association between coherence metric and topic quality. We therefore repeated the expert-driven evaluation protocol described in section 3.1. We trained one standard LDA model and one generalized Pólya urn model, each with $T = 200$, and randomly shuffled the 400 resulting topics. The topics were then presented to the experts from NINDS, with no indication as to the identity of the model from which each topic came. As these evaluations are time consuming, the experts evaluated the only the first 200 topics, which consisted of 103 generalized Pólya urn topics and 97 LDA topics. AUC values predicting bad topics given coherence were 0.83 and 0.80, respectively. Coherence effectively predicts topic quality in both models.

Although we were able to improve the average overall quality of topics and the average quality of the ten lowest-scoring topics, we found that the generalized Pólya urn model was less successful reducing the overall number of bad topics. Ignoring one "unbalanced" topic from each model, 16.5% of the LDA topics and 13.5% from the generalized Pólya urn model were marked as "bad." While this result is an improvement, it is not significant at $p = 0.05$.

6 Discussion

We have demonstrated the following:

- There is a class of low-quality topics that cannot be detected using existing word-intrusion tests, but that can be identified reliably using a metric based on word co-occurrence statistics.
- It is possible to improve the coherence score of topics, both overall and for the ten worst, while retaining the ability to flag bad topics, all without requiring semi-supervised data or additional reference corpora. Although additional information may be useful, it is not necessary.
- Such models achieve better performance with substantially fewer Gibbs iterations than LDA.

We believe that the most important challenges in future topic modeling research are improving the semantic quality of topics, particularly at the low end, and scaling to ever-larger data sets while ensuring high-quality topics. Our results provide critical insight into these problems. We found that it should be possible to construct unsupervised topic models that do not produce bad topics. We also found that Gibbs sampling mixes faster for models that use word co-occurrence information, suggesting that such methods may also be useful in guiding online stochastic variational inference (Hoffman et al., 2010).

Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval, in part by the CIA, the NSA and the NSF under NSF grant # IIS-0326249, in part by NIH:HHSN271200900640P, and in part by NSF # number SBE-0965436. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

Loulwah AlSumait, Daniel Barbara, James Gentle, and Carlotta Domeniconi. 2009. Topic significance ranking of LDA generative models. In *ECML*.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 25–32.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January.

K.R. Canini, L. Shi, and T.L. Griffiths. 2009. Online inference of topics with latent Dirichlet allocation. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 22*, pages 288–296.

Kenneth Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 6(1):22–29.

Gabriel Doyle and Charles Elkan. 2009. Accounting for burstiness in topic models. In *ICML*.

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transaction on Pattern Analysis and Machine Intelligence 6*, pages 721–741.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl. 1):5228–5235.

Matthew Hoffman, David Blei, and Francis Bach. 2010. Online learning for latent dirichlet allocation. In *NIPS*.

Hosan Mahmoud. 2008. *Pólya Urn Models*. Chapman & Hall/CRC Texts in Statistical Science.

Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 490–499.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Yee Whye Teh, Dave Newman, and Max Welling. 2006. A collapsed variational Bayesian inference algorithm for lat ent Dirichlet allocation. In *Advances in Neural Information Processing Systems 18*.

Hanna Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *Proceedings of the 26th Interational Conference on Machine Learning*.

Xing Wei and Bruce Croft. 2006. LDA-based document models for ad-hoc retrival. In *Proceedings of the 29th Annual International SIGIR Conference*.

A Weakly-supervised Approach to Argumentative Zoning of Scientific Documents

Yufan Guo
Computer Laboratory
University of Cambridge, UK
yg244@cam.ac.uk

Anna Korhonen
Computer Laboratory
University of Cambridge, UK
alk23@cam.ac.uk

Thierry Poibeau
LaTTiCe, UMR8094
CNRS & ENS, France
thierry.poibeau@ens.fr

Abstract

Argumentative Zoning (AZ) – analysis of the argumentative structure of a scientific paper – has proved useful for a number of information access tasks. Current approaches to AZ rely on supervised machine learning (ML). Requiring large amounts of annotated data, these approaches are expensive to develop and port to different domains and tasks. A potential solution to this problem is to use weakly-supervised ML instead. We investigate the performance of four weakly-supervised classifiers on scientific abstract data annotated for multiple AZ classes. Our best classifier based on the combination of active learning and self-training outperforms our best supervised classifier, yielding a high accuracy of 81% when using just 10% of the labeled data. This result suggests that weakly-supervised learning could be employed to improve the practical applicability and portability of AZ across different information access tasks.

1 Introduction

Many practical tasks require accessing specific types of information in scientific literature. For example, a reader of scientific literature may be looking for information about the objective of the study in question, the methods used in the study, the results obtained, or the conclusions drawn by authors. Similarly, many Natural Language Processing (NLP) tasks focus on the extraction of specific types of information in documents only.

To date, a number of approaches have been proposed for sentence-based classification of scientific literature according to categories of information structure (or discourse, rhetorical, argumentative or conceptual structure, depending on the framework in question). Some of these classify sentences according to typical section names seen in scientific documents (Lin et al., 2006; Hirohata et al., 2008), while others are based e.g. on argumentative zones (Teufel and Moens, 2002; Mizuta et al., 2006; Teufel et al., 2009), qualitative dimensions (Shatkay et al., 2008) or conceptual structure (Liakata et al., 2010) of documents.

The best of current approaches have yielded promising results and proved useful for information retrieval, information extraction and summarization tasks (Teufel and Moens, 2002; Mizuta et al., 2006; Tbahriti et al., 2006; Ruch et al., 2007). However, relying on fully supervised machine learning (ML) and a large body of annotated data, existing approaches are expensive to develop and port to different scientific domains and tasks.

A potential solution to this bottleneck is to develop techniques based on weakly-supervised ML. Relying on a small amount of labeled data and a large pool of unlabeled data, weakly-supervised techniques (e.g. semi-supervision, active learning, co/tri-training, self-training) aim to keep the advantages of fully supervised approaches. They have been applied to a wide range of NLP tasks, including named-entity recognition, question answering, information extraction, text classification and many others (Abney, 2008), yielding performance levels similar or equivalent to those of fully supervised techniques.

To the best of our knowledge, such techniques

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 273–283,
Edinburgh, Scotland, UK, July 27–31, 2011.

have not yet been applied to the analysis of information structure of scientific documents by aforementioned approaches. Recent experiments have demonstrated the usefulness of weakly-supervised learning for classifying discourse relations in scientific texts, e.g. (Hernault et al., 2011). However, focusing on local (rather than global) structure of documents and being much more fine-grained in nature, this related task differs from ours considerably.

In this paper, we investigate the potential of weakly-supervised learning for Argumentative Zoning (AZ) of scientific abstracts. AZ is an approach to information structure which provides an analysis of the rhetorical progression of the scientific argument in a document (Teufel and Moens, 2002). It has been used to analyze scientific texts in various disciplines – including computational linguistics (Teufel and Moens, 2002), law, (Hachey and Grover, 2006), biology (Mizuta et al., 2006) and chemistry (Teufel et al., 2009) – and has proved useful for NLP tasks such as summarization (Teufel and Moens, 2002). Although the basic scheme is said to be discipline-independent (Teufel et al., 2009), its application to different domains has resulted in various modifications and laborious annotation exercises. This suggests that a weakly-supervised approach would be more practical than a fully supervised one for the real-world application of AZ.

Taking two supervised classifiers as a comparison point – Support Vector Machines (SVM) and Conditional Random Fields (CRF) – we investigate the performance of four weakly-supervised classifiers on the AZ task: two based on semi-supervised learning (transductive SVM and semi-supervised CRF) and two on active learning (Active SVM alone and in combination with self-training).

The results are promising. Our best weakly-supervised classifier (Active SVM with self-training) outperforms the best supervised classifier (SVM), yielding high accuracy of 81% when using just 10% of the labeled data. When using just one third of the labeled data, it performs equally well as a fully supervised SVM which uses 100% of the labeled data. Our investigation suggests that weakly-supervised learning could be employed to improve the practical applicability and portability of AZ to different information access tasks.

2 Data

We used in our experiments the recent dataset of (Guo et al., 2010). Guo et al. (2010) provide a corpus of 1000 biomedical abstracts (consisting of 7985 sentences and 225785 words) annotated according to three schemes of information structure – those based on section names (Hirohata et al., 2008), AZ (Mizuta et al., 2006) and Core Scientific Concepts (CoreSC) (Liakata et al., 2010). We focus here on AZ only, because it subsumes all the categories of the simple section name -based scheme, and according to the inter-annotator agreement and ML experiments reported by Guo et al. (2010) it performs better on this data than the fairly fine-grained CoreSC scheme.

AZ is a scheme which provides an analysis of the rhetorical progression of the scientific argument, following the knowledge claims made by authors. (Teufel and Moens, 2002) introduced AZ and applied it first to computational linguistics papers. (Hachey and Grover, 2006) applied the scheme later to legal texts and (Mizuta et al., 2006) modified it for biology papers. More recently, (Teufel et al., 2009) introduced a refined version of AZ and applied it to chemistry papers.

The biomedical dataset of (Guo et al., 2010) has been annotated according to the version of AZ developed for biology papers (Mizuta et al., 2006) (with only minor modifications concerning zone names). Seven categories of this scheme (out of the 10 possible) actually appear in abstracts and in the resulting corpus. These are shown and explained in Table 1. For example, the Method zone (METH) is for sentences which describe *a way of doing research, esp. according to a defined and regular plan; a special form of procedure or characteristic set of procedures employed in a field of study as a mode of investigation and inquiry*.

An example of a biomedical abstract annotated according to AZ is shown in Figure 1, with different zones highlighted in different colors. For example, the RES zone is highlighted in lemon green.

Table 2 shows the distribution of sentences per scheme category in the corpus: Results (RES) is by far the most frequent zone (accounting for 40% of the corpus), while Background (BKG), Objective (OBJ), Method (METH) and Conclusion (CON) cover

Table 1: Categories of AZ appearing in the corpus of (Guo et al., 2010)

Category	Abbr.	Definition
Background	BKG	The circumstances pertaining to the current work, situation, or its causes, history, etc.
Objective	OBJ	A thing aimed at or sought, a target or goal
Method	METH	A way of doing research, esp. according to a defined and regular plan; a special form of procedure or characteristic set of procedures employed in a field of study as a mode of investigation and inquiry
Result	RES	The effect, consequence, issue or outcome of an experiment; the quantity, formula, etc. obtained by calculation
Conclusion	CON	A judgment or statement arrived at by any reasoning process; an inference, deduction, induction; a proposition deduced by reasoning from other propositions; the result of a discussion, or examination of a question, final determination, decision, resolution, final arrangement or agreement
Related work	REL	A comparison between the current work and the related work
Future work	FUT	The work that needs to be done in the future

Figure 1: An example of an annotated abstract

Butadiene (BD) metabolism shows gender, species and concentration dependency, making the extrapolation of animal results to humans complex. BD is metabolized mainly by cytochrome P450 2E1 to three epoxides, 1,2-epoxy-3-butene (EB), 1,2;3,4-diepoxybutane (DEB) and 1,2-epoxy-butanediol (EB-diol). For accurate risk assessment it is important to elucidate species differences in the internal formation of the individual epoxides in order to assign the relative risks associated with their different mutagenic potencies. Analysis of N-terminal globin adducts is a common approach for monitoring the internal formation of BD ***Background***es. Our long term strategy is to develop an LC-MS/MS method for simultaneous detection of all three BD hemoglobin adducts. This approach is modeled after the recently reported immunoaffinity LC-MS/MS method for the cyclic N,N-(2,3-dihydroxy-1,4-butadyil)-valine (pyr-Val, derived from DEB). We report herein the analysis of the EB-derived 2-hydroxyl-3-butenyl-valine pep ***Objective***l). The procedure utilizes trypsin hydrolysis of globin and immunoaffinity (IA) purification of alkylated heptapeptides. Quantitation is based on LC-MS/MS monitoring of the transition from the singly charged molecular ion of HB-Val (1-7) to the a(1) fragment. Human HB-Val (1-11) was synthesized and used for antibody production. As internal standard, the labeled rat-[(13)C(5)(15)N]-Val (1-11) was prepared through direct alkylation of the corresponding peptide with EB. Standards were characterized and quantified by LC-MS/MS and LC-UV. The method was validated with different amounts of human HB-Val standard. The recovery was >75% and coefficient of variation <25%. The LOQ was set to 100 fmol/injection. For a proof of principal experiment, globin samples from male and female rats exposed to 1000 ppm BD for 90 days w***Method***ed. The amounts of HB-Val present were 268.2+/-56 and 350+/-70 pmol/g (mean+/-S.D.) for males and females, respectively. No HB-Val was detected i***Result***ls. These data are much lower compared to previously reported values measu***Related work***MS. The difference may be due higher specificity of the LC-MS/MS method to the N-terminal peptide from the alpha-chain versus derivatization of both alpha- and beta-chain by Edman degradation, and possible instability of HB-Val adducts during long term storage (about 10 years) betw***Conclusion***es. These differences will be resolved by examining recently collected samples, using the same internal standard for parallel analysis by GC-MS/M***Future work***MS. Based on our experience with pyr-Val adduct assay we anticipate that this assay will be suitable for evaluation of HB-Val in multiple species.

Table 2: Distribution of sentences in the AZ-annotated corpus

	BKG	OBJ	METH	RES	CON	REL	FUT
Word	36828	23493	41544	89538	30752	2456	1174
Sentence	1429	674	1473	3185	1082	95	47
Sentence	18%	8%	18%	40%	14%	1%	1%

8-18% of the corpus each. Two categories are very low in frequency, only covering 1% of the corpus each: Related work (REL) and Future work (FUT).

Guo et al. (2010) report the inter-annotator agreement between their three annotators: one linguist, one computational linguist and one domain expert. According to Cohen's kappa (Cohen, 1960) the agreement is relatively high: $\kappa = 0.85$.

3 Automatic identification of AZ

3.1 Features and feature extraction

Guo et al. (2010) used a variety of features in their fully supervised ML experiments on different schemes of information structure. Since their feature types cover the best performing feature types in earlier works e.g. (Teufel and Moens, 2002; Lin et al., 2006; Mullen et al., 2005; Hirohata et al., 2008; Merity et al., 2009) we re-implemented and used them in our experiment[1]. However, being aware of the fact that some of these features may not be optimal for weakly-supervised learning (i.e. when learning from smaller data), we evaluate their performance and suitability for the task later in section 4.3.

- **Location**. Zones tend to appear in typical positions in abstracts. Each abstract was there-

[1]The only exception is the history feature which was left out because it cannot be applied to all of our methods

fore divided into ten parts (1-10, measured by the number of words), and the location was defined by the parts where the sentence begins and ends.

- **Word**. All the words in the corpus.
- **Bi-gram**. Any combination of two adjacent words in the corpus.
- **Verb**. All the verbs in the corpus.
- **Verb Class**. 60 verb classes appearing in biomedical journal articles.
- **Part-of-Speech – POS**. The POS tag of each verb in the corpus.
- **Grammatical Relation – GR**. Subject (*ncsubj*), direct object (*dobj*), indirect object (*iobj*) and second object (*obj2*) relations in the corpus. e.g. *(ncsubj observed_14 difference_5 obj)*. The value of this feature equals 1 if it occurs in a particular sentence (and 0 if not).
- **Subj and Obj**. The subjects and objects appearing with any verbs in the corpus (extracted from above GRs).
- **Voice**. The voice of verbs (active or passive) in the corpus.

These features were extracted from the corpus using a number of tools. A tokenizer was used to detect the boundaries of sentences and to separate punctuation from adjacent words e.g. in complex biomedical terms such as *2-amino-3,8-diethylimidazo[4,5-f]quinoxaline*. The C&C tools (Curran et al., 2007) trained on biomedical literature were employed for POS tagging, lemmatization and parsing. The lemma output was used for creating Word, Bi-gram and Verb features. The GR output was used for creating the GR, Subj, Obj and Voice features. The "obj" marker in a subject relation indicates passive voice (e.g. *(ncsubj observed_14 difference_5 obj)*). The verb classes were acquired automatically from the corpus using the unsupervised spectral clustering method of (Sun and Korhonen, 2009). To control the number of features we lemmatized the lexical items for all the features, and removed the words and GRs with fewer than 2 occurrences and bi-grams with fewer than 5 occurrences.

3.2 Machine learning methods

Support Vector Machines (SVM) and Conditional Random Fields (CRF) have proved the best performing fully supervised methods in most recent works on information structure, e.g. (Teufel and Moens, 2002; Mullen et al., 2005; Hirohata et al., 2008; Guo et al., 2010). We therefore implemented these methods as well as weakly supervised variations of them: active SVM with and without self-training, transductive SVM and semi-supervised CRF.

3.2.1 Supervised methods

SVM constructs hyperplanes in a multidimensional space to separate data points of different classes. Good separation is achieved by the hyperplane that has the largest distance from the nearest data points of any class. The hyperplane has the form $w \cdot x - b = 0$, where w is its normal vector. We want to maximize the distance from the hyperplane to the data points, or the distance between two parallel hyperplanes each of which separates the data. The parallel hyperplanes can be written as: $w \cdot x - b = 1$ and $w \cdot x - b = -1$, and the distance between them is $\frac{2}{|w|}$. The problem reduces to:

Minimize $|w|$ (in w, b)

Subject to

$w \cdot x - b \geq 1$ for x of one class,

$w \cdot x - b \leq -1$ for x of the other,

which can be solved by using the SMO algorithm (Platt, 1999b). We used Weka software (Hall et al., 2009) (employing its linear kernel) for SVM experiments.

CRF is an undirected graphical model which defines a probability distribution over the hidden states (e.g. label sequences) given the observations. The probability of a label sequence y given an observation sequence x can be written as:

$p(y|x, \theta) = \frac{1}{Z(x)} exp(\sum_j \theta_j F_j(y, x))$,

where $F_j(y, x)$ is a real-valued feature function of the states and the observations; θ_j is the weight of F_j, and $Z(x)$ is a normalization factor. The θ parameters can be learned using the L-BFGS algorithm (Nocedal, 1980). We used Mallet software (McCallum, 2002) for CRF experiments.

3.2.2 Weakly-supervised methods

Active SVM (ASVM) starts with a small amount of labeled data, and iteratively chooses a proportion of

unlabeled data for which SVM has less confidence to be labeled (the labels can be restored from the original corpus) and used in the next round of learning, i.e. active learning. Query strategies based on the structure of SVM are frequently employed (Tong and Koller, 2001; Novak et al., 2006). For example, it is often assumed that the data points close to the separating hyperplane are those that the SVM is uncertain about. Unlike these methods, our learning algorithm compares the posterior probabilities of the best estimate given each unlabeled instance, and queries those with the lowest probabilities for the next round of learning. The probabilities can be obtained by fitting a Sigmoid after the standard SVM (Platt, 1999a), and combined using a pairwise coupling algorithm (Hastie and Tibshirani, 1998) in the multi-class case. We used the SVM linear kernel in Weka for classification, and the -M flag in Weka for calculating the posterior probabilities.

Active SVM with self-training (ASSVM) is an extension of ASVM where each round of training has two steps: (i) training on the labeled, and testing on the unlabeled data, and querying; (ii) training on both labeled and unlabeled/machine-labeled data by using the estimates from step (i). The idea of ASSVM is to make the best use of the labeled data, and to make the most use of the unlabeled data.

Transductive SVM (TSVM) is an extension of SVM which takes advantage of both labeled and unlabeled data (Vapnik, 1998). Similar to SVM, the problem is defined as:

Minimize $|w|$ (in $w, b, y^{(u)}$)
Subject to
$y^{(l)}(w \cdot x^{(l)} - b) \geq 1$,
$y^{(u)}(w \cdot x^{(u)} - b) \geq 1$,
$y^{(u)} \in \{-1, 1\}$,

where $x^{(u)}$ is unlabeled data and $y^{(u)}$ the estimate of its label. The problem can be solved by using the CCCP algorithm (Collobert et al., 2006). We used UniverSVM software (Sinz, 2011) for TSVM experiments.

Semi-supervised CRF (SSCRF) can be implemented with entropy regularization (ER). It extends the objective function on Labeled data $\sum_L \log p(y^{(l)}|x^{(l)}, \theta)$ with an additional term $\sum_U \sum_Y p(y|x^{(u)}, \theta) \log p(y|x^{(u)}, \theta)$ to minimize the conditional entropy of the model's predictions on Unlabeled data (Jiao et al., 2006; Mann and Mccallum, 2007). We used Mallet software (McCallum, 2002) for SSCRF experiments.

4 Experimental evaluation

4.1 Evaluation methods

We evaluated the ML results in terms of accuracy, precision, recall, and F-measure against manual AZ annotations in the corpus:

$$acc = \frac{no.\ of\ correctly\ classified\ sentences}{total\ no.\ of sentences\ in\ the\ corpus}$$

$$p = \frac{no.\ of\ sentences\ correctly\ identified\ as\ Class_i}{total\ no.\ of\ sentences\ identified\ as\ Class_i}$$

$$r = \frac{no.\ of\ sentences\ correctly\ identified\ as\ Class_i}{total\ no.\ of\ sentences\ in\ Class_i}$$

$$f = \frac{2*p*r}{p+r}$$

We used 10-fold cross validation for all the methods to avoid the possible bias introduced by relying on any particular split of the data. More specifically, the data was randomly assigned to ten folds of roughly the same size. Each fold was used once as test data and the remaining nine folds as training data. The results were then averaged.

Following (Dietterich, 1998), we used McNemar's test (McNemar, 1947) to measure the statistical significance between the results of different ML methods. The chosen significance level was .05.

4.2 Results

Table 3 shows the results for the four weakly-supervised and two supervised methods when 10% of the training data (i.e. ∼700 sentences) has been labeled. We can see that ASSVM is the best performing method with an accuracy of 81% and the macro

Table 3: Results when using 10% of the labeled data

	Acc.	F-score							
		MF	BKG	OBJ	METH	RES	CON	REL	FUT
SVM	.77	.74	.84	.68	.71	.82	.64	-	-
CRF	.70	.65	.75	.46	.48	.78	.76	-	-
ASVM	.80	.75	.88	.56	.68	.87	.78	.33	
ASSVM	.81	.76	.86	.56	.76	.88	.76	-	-
TSVM	.76	.73	.84	.61	.71	.79	.71	-	-
SSCRF	.73	.67	.76	.48	.52	.81	.78	-	-

MF: Macro F-score of the five high frequency categories: BKG, OBJ, METH, RES, CON.

Figure 2: Learning curve for different methods when using 0-100% of the labeled data

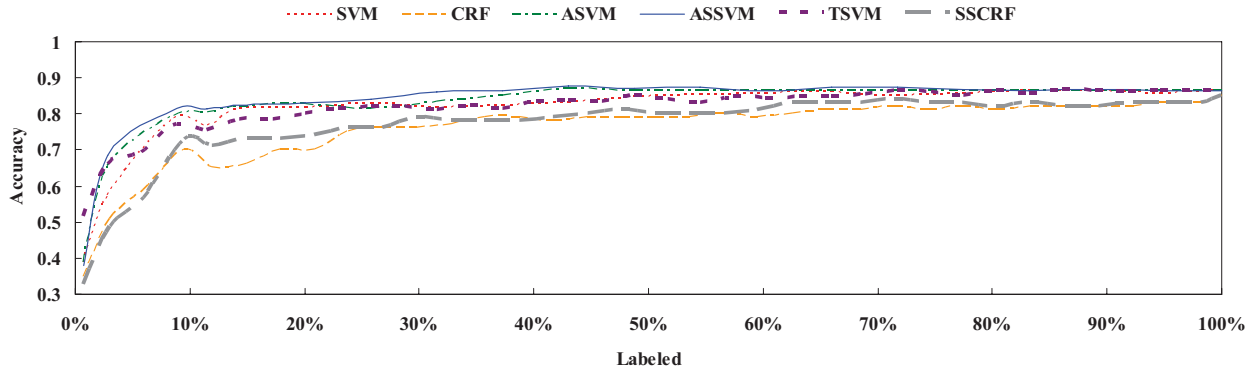
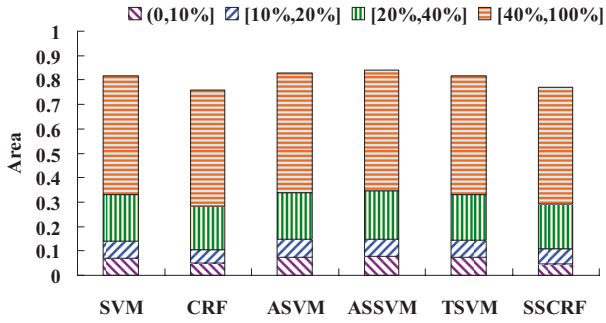


Figure 3: Area under learning curves at different intervals



F-score of .76 (the macro F-score is calculated for the 5 scheme categories which are found by all the methods). ASVM performs nearly as well, with an accuracy of 80% and F-score of .75. Both methods outperform supervised SVM with a statistically significant difference ($p < .001$).

TSVM is the lowest performing SVM-based method. Yielding an accuracy of 76% and F-score of .73 its performance is lower than that of the supervised SVM. However, it does outperform both CRF-based methods. SSCRF performs better than CRF with 3% higher accuracy and .02 higher F-score. The difference in accuracy is statistically significant ($p < .001$).

Only one method (ASVM) identifies six out of the seven possible categories. Other methods identify five categories. The 1-2 missing categories are very low in frequency (accounting for 1% of the corpus data each, see table 2). Looking at the results for other categories, they seem to reflect the amount of corpus data available for each category (Table 2), with RES (Results) being the highest and OBJ (Objective) the lowest performing category with most methods. Interestingly, the only method that performs relatively well on OBJ is the supervised SVM.

The best method ASSVM outperforms other methods most clearly on METH (Method) category. Although METH is a high frequency category (accounting for 18% of the corpus data) other methods tend to confuse it with OBJ, presumably because a single sentence may contain elements of both (e.g. scientists may describe some of their method when describing the objective of the study).

Figure 2 shows the learning curve of different methods (in terms of accuracy) when the percentage of the labeled data (in the training set) ranges from 0 to 100%. ASSVM outperforms other methods, reaching its best performance of 88% accuracy when using ∼40% of the labeled data. Indeed when using 33% of the labeled data, it performs already equally well as fully-supervised SVM using 100% of the labeled data. The advantage of ASSVM over ASVM (the second best method) is clear especially when 20-40% of the labeled data is used. SVM and TSVM tend to perform quite similarly with each other when more than 25% of the labeled data is used, but when less data is available, SVM performs better. Looking at the CRF-based methods, SSCRF outperforms CRF in particular when 10-25% of the labeled data is used. However, neither of them reaches the performance level of SVM-based methods.

Figure 3 shows the area under the learning curves (by the trapezoidal rule) at different intervals, which gives a reasonable approximation to the overall performance of different methods. The area under ASSVM is the largest at each of the four intervals, with a value of .08 at (0,10%], .07 at [10%,20%],

.20 at [20%, 40%] and .50 at [40%,100%]. The difference between supervised and weakly-supervised methods is more significant at (0, 20%] than at [20%,100%].

4.3 Further analysis of the features

As explained in section 3.1, we employed in our experiments a collection of features which had performed well in previous supervised AZ experiments. We conducted further analysis to investigate which of these features are the most (and the least) useful for weakly-supervised learning. We took our best performing method ASSVM and conducted leave-one-out analysis of the features with 10% of the labeled data. The results are shown in Table 4.

Table 4: Leaving one feature out results for ASSVM when using 10% of the labeled data

	Acc.	F-score							
		MF	BKG	OBJ	METH	RES	CON	REL	FUT
Location	.73	.67	.67	.55	.62	.85	.65	-	-
Word	.80	.78	.87	.70	.74	.85	.72	-	-
Bigram	.81	.75	.83	.57	.71	.87	.78	.33	-
Verb	.81	.79	.84	.77	.73	.87	.75	-	-
VC	.79	.75	.86	.62	.72	.84	.70	-	-
POS	.74	.70	.66	.65	.66	.82	.73	-	-
GR	.79	.75	.83	.67	.69	.84	.72	-	-
Subj	.80	.76	.87	.65	.73	.85	.72	-	-
Obj	.80	.78	.84	.75	.70	.85	.75	-	-
Voice	.78	.75	.88	.70	.71	.83	.62	-	-
Φ	.81	.76	.86	.56	.76	.88	.76	-	-

MF: Macro F-score of the five high frequency categories: BKG, OBJ, METH, RES, CON.
Φ: Employing all the features.

We can see that the Location feature is by far the most useful feature for ASSVM. The performance drops 8% in accuracy and .09 in F-score in the absence of this feature. Location is particularly important for BKG (which nearly always appears in the same location: in the beginning of an abstract) and is highly useful for METH and CON as well. Removing POS has almost equally strong effect, in particular on BKG and METH, suggesting that verb tense is particularly useful for distinguishing these categories.

Also Voice, Verb class and GR contribute to general performance, especially to accuracy. Voice is particularly important for CON, which differs from other categories in the sense that it is marked by frequent usage of active voice. Verb class is helpful for METH, RES and CON while GR is helpful for all high frequency categories.

Among the least helpful features are those which suffer from sparse data problems, including e.g. Word, Bi-gram, and Verb. They perform particularly badly when applied to low frequency zones. However, this is not the case when using fully-supervised methods (i.e. 100% of the labeled data), suggesting that a good performance in fully supervised experiments does not necessarily translate into a good performance in weakly-supervised experiments, and that careful feature analysis and selection is important when aiming to optimize the performance when learning from sparse data.

5 Discussion

In our experiments, the majority of weakly-supervised methods outperformed their corresponding supervised methods when using just 10% of the labeled data. The SVM-based methods performed better than the CRF-based ones (regardless of whether they were weakly or fully supervised). Guo et al. (2010) made a similar discovery when comparing fully supervised versions of SVM and CRF.

Our best performing weakly-supervised methods were those based on active learning. Making a good use of both labeled and unlabeled data, active learning combined with self-training (ASSVM) proved to be the most useful method. Given 10% of the labeled data, ASSVM obtained an accuracy of 81% and F-score of .76, outperforming the best supervised method SVM with a statistically significant difference. It reached its top performance (88% accuracy) when using 40% of the labeled data, and performed equally well as fully supervised SVM (i.e. 100% of the labeled data) when using just one third of the labeled data.

This result is in line with the results of many other text classification works where active learning (alone or in combination with other techniques such as self-training) has proved similarly useful, e.g. (Lewis and Gale, 1994; Tong and Koller, 2002; Brinker, 2006; Novak et al., 2006; Esuli and Sebastiani, 2009; Yang et al., 2009).

While active learning iteratively explores the unknown aspects of the unlabeled data, semi-supervised learning attempts to make the best use

of what it already knows about the data. In our experiments, semi-supervised methods (TSVM and SSCRF) did not perform equally well as active learning – TSVM even produced a lower accuracy than SVM with the same amount of labeled data – although these methods have gained success in related works.

We therefore looked into related works using TSVM, e.g. (Chapelle and Zien, 2005), and discovered that our dataset is much higher in dimensionality than those employed in many other works. High dimensional data is more sensitive, and therefore fine-tuning with unlabeled data may cause a big deviation. We also looked into related works using SSCRF, in particular the work of (Jiao et al., 2006) who used the same SSCRF as the one we used in our experiments. Jiao et al. (2006) employed a much larger data set than we did – one including 5448 labeled instances (in 3 classes) and 5210-25145 unlabeled instances. Given more labeled and unlabeled data per class we might be able to obtain better performance using SSCRF also on our task. However, given the high cost of obtaining labeled data methods not needing it are preferable.

6 Conclusions and future work

Our experiments show that weakly-supervised learning can be used to identify AZ in scientific documents with good accuracy when only a limited amount of labeled data is available. This is helpful thinking of the real-world application and porting of the approach to different tasks and domains. To the best of our knowledge, no previous work has been done on weakly-supervised learning of information structure according to schemes of the type we have focused on (Teufel and Moens, 2002; Mizuta et al., 2006; Lin et al., 2006; Hirohata et al., 2008; Shatkay et al., 2008; Liakata et al., 2010).

Recently, some work has been done on the related task of classification of discourse relations in scientific texts: (Hernault et al., 2011) used structural learning (Ando and Zhang, 2005) for this task. They obtained 30-60% accuracy on the RST Discourse Treebank (including 41 relation types) when using 100-10000 labeled and 100000 unlabeled instances. The accuracy was 20-60% when using the labeled data only. However, although related, the task of discourse relation classification differs substantially from our task in that it focuses on local discourse relations while our task focuses on the global structure of the scientific document.

In the future, we plan to improve and extend this work in several directions. First, the approach to active learning could be improved in various ways. The query strategy we employed (uncertainty sampling) is a relatively straightforward method which only considers the best estimate for each unlabeled instance, disregarding other estimates that may contain useful information. In the future, we plan to experiment with more sophisticated strategies, e.g. the margin sampling algorithm by (Scheffer et al., 2001) and the query-by-committee (QBC) algorithm by (Seung et al., 1992). In addition, there are algorithms designed for reducing the redundancy in queries which may be worth investigating (Hoi et al., 2006).

Also, (Hoi et al., 2006) shows that Logistic Regression (LR) outperforms SVM when used with active learning, yielding higher F-score on the Reuters-21578 data set (binary classification, 10,788 documents in total, 100 of them labeled). It would be interesting to explore whether supervised methods other than SVM are optimal for active learning when applied to our task.

Secondly, we plan to investigate other semi-supervised methods, for example, the Expectation-Maximization (EM) algorithm. (Lanquillon, 2000) has shown that EM SVM performs better than supervised and transductive SVM on a text classification task when applied to the dataset of 20 Newsgroups (20 classes, 4000 documents for testing, 10000 unlabeled ones), yielding up to $\sim$10% higher accuracy when 200-5000 labeled documents are used for training.

In addition, other combinations of weakly-supervised methods might be worth looking into, such as EM+active learning (McCallum and Nigam, 1998) and co-training+EM+active learning (Muslea et al., 2002), which have proved promising in related text classification works.

Besides looking for optimal ML strategies, we plan to look for optimal features for the task. Our feature analysis showed that not all the features which had proved promising in fully supervised experiments were equally promising when applied to weakly-supervised learning from smaller data. We

plan to look into ways of reducing the sparse data problem in features, e.g. by classifying not only verbs but also other word classes into semantically-motivated categories.

One the key motivations for developing a weakly-supervised approach is to facilitate easy porting of schemes such as AZ to new tasks and domains. Recent research shows that active learning in a target domain can leverage information from a different but related (source) domain (Rai et al., 2010). Making use of existing annotated datasets in biology, chemistry, computational linguistics and law (Teufel and Moens, 2002; Mizuta et al., 2006; Hachey and Grover, 2006; Teufel et al., 2009) we will explore optimal ways of combining weakly-supervised learning with domain-adaptation.

The work presented in this paper has focused on the abstracts annotated according to the AZ scheme. In the future, we plan to investigate the usefulness of weakly-supervised learning for identifying other schemes of information structure, e.g. (Lin et al., 2006; Hirohata et al., 2008; Shatkay et al., 2008; Liakata et al., 2010), and not only in scientific abstracts but also in full journal papers which typically exemplify a larger set of scheme categories.

Finally, an important avenue of future research is to evaluate the usefulness of weakly-supervised identification of information structure for NLP tasks such as summarization and information extraction (Tbahriti et al., 2006; Ruch et al., 2007), and for practical tasks such as manual review of scientific papers for research purposes (Guo et al., 2010).

Acknowledgments

The work reported in this paper was funded by the Royal Society (UK). YG was funded by the Cambridge International Scholarship.

References

Steven Abney. 2008. *Semi-supervised learning for computational linguistics*. Chapman & Hall / CRC.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853.

Klaus Brinker. 2006. On active learning in multi-label classification. In *From Data and Information Analysis to Knowledge Engineering*, pages 206–213.

Olivier Chapelle and Alexander Zien. 2005. Semi-supervised classification by low density separation.

J. Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. 2006. Trading convexity for scalability. In *Proceedings of the 23rd international conference on Machine learning*.

J. R. Curran, S. Clark, and J. Bos. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the ACL 2007 Demonstrations Session*.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10:1895–1923.

Andrea Esuli and Fabrizio Sebastiani. 2009. Active learning strategies for multi-label text classification. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*.

Yufan Guo, Anna Korhonen, Maria Liakata, Ilona Silins Karolinska, Lin Sun, and Ulla Stenius. 2010. Identifying the information structure of scientific abstracts: an investigation of three different schemes. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*.

Ben Hachey and Claire Grover. 2006. Extractive summarisation of legal texts. *Artif. Intell. Law*, 14:305–345.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18.

T. Hastie and R. Tibshirani. 1998. Classification by pairwise coupling. *Advances in Neural Information Processing Systems*, 10.

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. 2011. Semi-supervised discourse relation classification with structural learning. In *CICLing (1)*.

K. Hirohata, N. Okazaki, S. Ananiadou, and M. Ishizuka. 2008. Identifying sections in scientific abstracts using conditional random fields. In *Proceedings of 3rd International Joint Conference on Natural Language Processing*.

Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. 2006. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th international conference on World Wide Web*.

F. Jiao, S. Wang, C. Lee, R. Greiner, and D. Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *COLING/ACL*.

Carsten Lanquillon. 2000. Learning from labeled and unlabeled documents: A comparative study on semi-supervised text classification. In *Proceedings of the*

4th European Conference on Principles of Data Mining and Knowledge Discovery.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*.

M. Liakata, S. Teufel, A. Siddharthan, and C. Batchelor. 2010. Corpora for the conceptualisation and zoning of scientific papers. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

J. Lin, D. Karakos, D. Demner-Fushman, and S. Khudanpur. 2006. Generative content models for structural analysis of medical abstracts. In *Proceedings of BioNLP-06*.

G. S. Mann and A. Mccallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *HLT-NAACL*.

Andrew McCallum and Kamal Nigam. 1998. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*.

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

Quinn McNemar. 1947. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157.

S. Merity, T. Murphy, and J. R. Curran. 2009. Accurate argumentative zoning with maximum entropy models. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*.

Y. Mizuta, A. Korhonen, T. Mullen, and N. Collier. 2006. Zone analysis in biology articles as a basis for information extraction. *International Journal of Medical Informatics on Natural Language Processing in Biomedicine and Its Applications*, 75(6):468–487.

T. Mullen, Y. Mizuta, and N. Collier. 2005. A baseline feature set for learning rhetorical zones using full articles in the biomedical domain. *Natural language processing and text mining*, 7(1):52–58.

Ion Muslea, Steven Minton, and Craig A. Knoblock. 2002. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*.

Jorge Nocedal. 1980. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151):773–782.

Bla Novak, Dunja Mladeni, and Marko Grobelnik. 2006. Text classification with active learning. In *From Data and Information Analysis to Knowledge Engineering*, pages 398–405.

J. C. Platt. 1999a. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classiers*, pages 61–74.

John C. Platt. 1999b. Using analytic qp and sparseness to speed training of support vector machines. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*.

Piyush Rai, Avishek Saha, Hal Daumé, III, and Suresh Venkatasubramanian. 2010. Domain adaptation meets active learning. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*.

P. Ruch, C. Boyer, C. Chichester, I. Tbahriti, A. Geissbuhler, P. Fabry, J. Gobeill, V. Pillet, D. Rebholz-Schuhmann, C. Lovis, and A. L. Veuthey. 2007. Using argumentation to extract key sentences from biomedical abstracts. *Int J Med Inform*, 76(2-3):195–200.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden markov models for information extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*.

H. Shatkay, F. Pan, A. Rzhetsky, and W. J. Wilbur. 2008. Multi-dimensional classification of biomedical text: Toward automated, practical provision of high-utility text to diverse users. *Bioinformatics*, 24(18):2086–2093.

F. Sinz, 2011. *UniverSVM Support Vector Machine with Large Scale CCCP Functionality*. http://www.kyb.mpg.de/bs/people/fabee/universvm.html.

L. Sun and A. Korhonen. 2009. Improving verb clustering with automatically acquired selectional preference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

I. Tbahriti, C. Chichester, Frederique Lisacek, and P. Ruch. 2006. Using argumentation to retrieve articles with similar citations. *Int J Med Inform*, 75(6):488–495.

S. Teufel and M. Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics*, 28:409–445.

S. Teufel, A. Siddharthan, and C. Batchelor. 2009. Towards domain-independent argumentative zoning: Evidence from chemistry and computational linguistics. In *Proceedings of EMNLP*.

S. Tong and D. Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66.

V. N. Vapnik. 1998. *Statistical learning theory*. Wiley, New York.

Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. 2009. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Linear Text Segmentation Using Affinity Propagation

Anna Kazantseva
School of Electrical Engineering
and Computer Science,
University of Ottawa
ankazant@site.uottawa.ca

Stan Szpakowicz
School of Electrical Engineering
and Computer Science,
University of Ottawa &
Institute of Computer Science,
Polish Academy of Sciences
szpak@site.uottawa.ca

Abstract

This paper presents a new algorithm for linear text segmentation. It is an adaptation of Affinity Propagation, a state-of-the-art clustering algorithm in the framework of factor graphs. Affinity Propagation for Segmentation, or *APS*, receives a set of pairwise similarities between data points and produces segment boundaries and segment *centres* – data points which best describe all other data points within the segment. *APS* iteratively passes messages in a cyclic factor graph, until convergence. Each iteration works with information on all available similarities, resulting in high-quality results. *APS* scales linearly for realistic segmentation tasks. We derive the algorithm from the original Affinity Propagation formulation, and evaluate its performance on topical text segmentation in comparison with two state-of-the art segmenters. The results suggest that *APS* performs on par with or outperforms these two very competitive baselines.

1 Introduction

In complex narratives, it is typical for the topic to shift continually. Some shifts are gradual, others – more abrupt. *Topical text segmentation* identifies the more noticeable topic shifts. A topical segmenter's output is a very simple picture of the document's structure. Segmentation is a useful intermediate step in such applications as subjectivity analysis (Stoyanov and Cardie, 2008), automatic summarization (Haghighi and Vanderwende, 2009), question answering (Oh, Myaeng, and Jang, 2007) and others. That is why improved quality of text segmentation can benefit other language-processing tasks.

We present Affinity Propagation for Segmentation (*APS*), an adaptation of a state-of-the-art clustering algorithm, Affinity Propagation (Frey and Dueck, 2007; Givoni and Frey, 2009).[1] The original AP algorithm considerably improved exemplar-based clustering both in terms of speed and the quality of solutions. That is why we chose to adapt it to segmentation. At its core, *APS* is suitable for segmenting any sequences of data, but we present it in the context of segmenting documents. *APS* takes as input a matrix of pairwise similarities between sentences and, for each sentence, a *preference* value which indicates an *a priori* belief in how likely a sentence is to be chosen as a segment centre. *APS* outputs segment assignments and segment centres – data points which best explain all other points in a segment. The algorithm attempts to maximize *net similarity* – the sum of similarities between all data points and their respective segment centres.

APS operates by iteratively passing messages in a factor graph (Kschischang, Frey, and Loeliger, 2001) until a good set of segments emerges. Each iteration considers all similarities – takes into account all available information. An iteration includes sending at most $O(N^2)$ messages. For the majority of realistic segmentation tasks, however, the upper bound is $O(MN)$ messages, where M is a constant. This is more computationally expensive than the requirements of locally informed segmentation algorithms such as those based on HMM or CRF (see Section 2), but for a globally-informed algorithm the requirements are very reasonable. *APS* is an instance of loopy-belief propagation (belief propagation on cyclic graphs) which has

[1] An implementation of *APS* in Java, and the data sets, can be downloaded at ⟨www.site.uottawa.ca/∼ankazant⟩.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 284–293,
Edinburgh, Scotland, UK, July 27–31, 2011.

been used to achieved state-of-the-art performance in error-correcting decoding, image processing and data compression. Theoretically, such algorithms are not guaranteed to converge or to maximize the objective function. Yet in practice they often achieve competitive results.

APS works on an already pre-compiled similaritiy matrix, so it offers flexibility in the choice of similarity metrics. The desired number of segments can be set by adjusting preferences.

We evaluate the performance of *APS* on three tasks: finding topical boundaries in transcripts of course lectures (Malioutov and Barzilay, 2006), identifying sections in medical textbooks (Eisenstein and Barzilay, 2008) and identifying chapter breaks in novels. We compare *APS* with two recent systems: the Minimum Cut segmenter (Malioutov and Barzilay, 2006) and the Bayesian segmenter (Eisenstein and Barzilay, 2008). The comparison is based on the WindowDiff metric (Pevzner and Hearst, 2002). *APS* matches or outperforms these very competitive baselines.

Section 2 of the paper outlines relevant research on topical text segmentation. Section 3 briefly covers the framework of factor graphs and outlines the original Affinity Propagation algorithm for clustering. Section 4 contains the derivation of the new update messages for APSeg. Section 5 describes the experimental setting, Section 6 reports the results, Section 7 discusses conclusions and future work.

2 Related Work

This sections discusses selected text segmentation methods and positions the proposed *APS* algorithm in that context.

Most research on automatic text segmentation revolves around a simple idea: when the topic shifts, so does the vocabulary (Youmans, 1991). We can roughly subdivide existing approaches into two categories: locally informed and globally informed.

Locally informed segmenters attempt to identify topic shifts by considering only a small portion of complete document. A classical approach is TextTiling (Hearst, 1997). It consists of sliding two adjacent windows through text and measuring lexical similarity between them. Drops in similarity correspond to topic shifts. Other examples include text segmentation using Hidden Markov Models (Blei and Moreno, 2001) or Conditional Random Fields (Lafferty, McCallum, and Pereira, 2001). Locally informed methods are often very efficient because of lean memory and CPU time requirements. Due to a limited view of the document, however, they can easily be thrown off by short inconsequential digressions in narration.

Globally informed methods consider "the big picture" when determining the most likely location of segment boundaries. Choi (2000) applies divisive clustering to segmentation. Malioutov and Barzilay (2006) show that the knowledge about long-range similarities between sentences improves segmentation quality. They cast segmentation as a graph-cutting problem. The document is represented as a graph: nodes are sentences and edges are weighted using a measure of lexical similarity. The graph is cut in a way which maximizes the net edge weight within each segment and minimizes the net weight of severed edges. Such *Minimum Cut* segmentation resembles *APS* the most among others mentioned in this paper. The main difference between the two is in different objective functions.

Another notable direction in text segmentation uses generative models to find segment boundaries. Eisenstein and Barzilay (2008) treat words in a sentence as draws from a multinomial language model. Segment boundaries are assigned so as to maximize the likelihood of observing the complete sequence. Misra et al. (2009) use a Latent Dirichlet allocation topic model (Blei, Ng, and Jordan, 2003) to find coherent segment boundaries. Such methods output segment boundaries and suggest lexical distribution associated with each segment. Generative models tend to perform well, but are less flexible than the similarity-based models when it comes to incorporating new kinds of information.

Globally informed models generally perform better, especially on more challenging datasets such as speech recordings, but they have – unsurprisingly – higher memory and CPU time requirements.

The *APS* algorithm described in this paper combines several desirable properties. It is unsupervised and, unlike most other segmenters, does not require specifying the desired number of segments as an input parameter. On each iteration it takes into account the information about a large portion of the docu-

ment (or all of it). Because *APS* operates on a precompiled matrix of pair-wise sentence similarities, it is easy to incorporate new kinds of information, such as synonymy or adjacency. It also provides some information as to what the segment is about, because each segment is associated with a segment centre.

3 Factor graphs and affinity propagation for clustering

3.1 Factor graphs and the max-sum algorithm

The *APS* algorithm is an instance of belief propagation on a cyclic factor graph. In order to explain the derivation of the algorithm, we will first briefly introduce factor graphs as a framework.

Many computational problems can be reduced to maximizing the value of a multi-variate function $F(x_1, \ldots, x_n)$ which can be approximated by a sum of simpler functions. In Equation 1, H is a set of discrete indices and f_h is a local function with arguments $X_h \subset \{x_1, \ldots, x_n\}$:

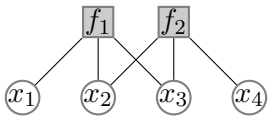
$$F(x_1, \ldots, x_n) = \sum_{h \in H} f_h(X_h) \tag{1}$$

Factor graphs offer a concise graphical representation for such problems. A global function F which can be decomposed into a sum of M local function f_h can be represented as a bi-partite graph with M function nodes and N variable nodes ($M = |H|$). Figure 1 shows an example of a factor graph for $F(x_1, x_2, x_3, x_4) = f_1(x_1, x_2, x_3) + f_2(x_2, x_3, x_4)$. The factor (or function) nodes are dark squares, the variable nodes are light circles.

The well-known max-sum algorithm (Bishop, 2006) seeks a configuration of variables which maximizes the objective function. It finds the maximum in acyclic factor graphs, but in graphs with cycles neither convergence nor optimality are guaranteed (Pearl, 1982). Yet in practice good approximations can be achieved. The max-sum algorithm amounts to propagating messages from function nodes to variable nodes and from variable nodes to function nodes. A message sent from a variable node x to a function node f is computed as a sum of the incoming messages from all neighbours of x other than f (the sum is computed for each possible value of x):

$$\mu_{x \to f} = \sum_{f' \in N(x) \setminus f} \mu_{f' \to x} \tag{2}$$

Figure 1: Factor graph for $F(x_1, x_2, x_3, x_4) = f_1(x_1, x_2, x_3) + f_2(x_2, x_3, x_4)$.



$N(x)$ is the set of all function nodes which are x's neighbours. The message reflects the evidence about the distribution of x from all functions which have x as an argument, except for the function corresponding to the receiving node f.

A message $\mu_{f \to x}$ from function f to variable x is computed as follows:

$$\mu_{f \to x} = \max_{N(f) \setminus x} (f(x_1, \ldots, x_m) + \sum_{x' \in N(f) \setminus x} \mu_{x' \to f}) \tag{3}$$

$N(f)$ is the set of all variable nodes which are f's neighbours. The message reflects the evidence about the distribution of x from function f and its neighbours other than x.

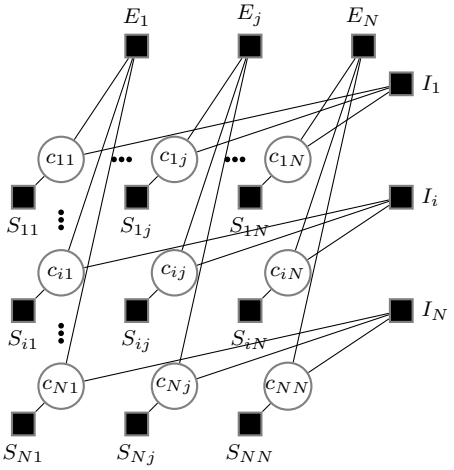
A common message-passing schedule on cyclic factor graphs is *flooding*: iteratively passing all variable-to-function messages, then all function-to-variable messages. Upon convergence, the summary message reflecting final beliefs about the maximizing configuration of variables is computed as a sum of all incoming function-to-variable messages.

3.2 Affinity Propagation

The *APS* algorithm described in this paper is a modification of the original Affinity Propagation algorithm intended for exemplar-based clustering (Frey and Dueck, 2007; Givoni and Frey, 2009). This section describes the binary variable formulation proposed by Givoni and Frey, and lays the groundwork for deriving the new update messages (Section 4).

Affinity Propagation for exemplar-based clustering is formulated as follows: to cluster N data points, one must specify a matrix of pairwise similarities $\{\mathcal{SIM}(i,j)\}_{i,j \in \{1,\ldots,N\}, i \neq j}$ and a set of self-similarities (so-called *preferences*) $\mathcal{SIM}(j,j)$ which reflect *a priori* beliefs in how likely each data point is to be selected as an exemplar. Preference values occupy the diagonal of the similarity matrix. The algorithm then assigns each data point to an exemplar so as to maximize net similarity – the sum of

Figure 2: Factor graph for affinity propagation.



similarities between all points and their respective exemplars; this is expressed by Equation 7. Figure 2 shows a schematic factor graph for this problem, with N^2 binary variables. $c_{ij} = 1$ iff point j is an exemplar for point i. Function nodes E_j enforce a *coherence constraint*: a data point cannot exemplify another point unless it is an exemplar for itself:

$$E_j(c_{1j}, \ldots, c_{Nj}) = \begin{cases} -\infty & \text{if } c_{jj} = 0 \ \wedge \ c_{ij} = 1 \\ & \text{for some } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

An I node encodes a *single-cluster constraint*: each data point must belong to exactly one exemplar – and therefore to one cluster:

$$I_i(c_{i1}, \ldots, c_{iN}) = \begin{cases} -\infty & \text{if } \sum_j c_{ij} \neq 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

An S node encodes user-defined similarities between data-points and candidate exemplars ($\mathcal{SIM}(i,j)$ is the similarity between points i and j):

$$S_{ij}(c_{ij}) = \begin{cases} \mathcal{SIM}(i,j) & \text{if } c_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Equation 7 shows the objective function which we want to maximize: a sum of similarities between data points and their exemplars, subject to the two constraints (coherence and single-cluster per point).

$$S(c_{11}, \ldots, c_{NN}) = \sum_{i,j} S_{i,j}(c_{ij}) + \sum_i I_i(c_{i1}, \ldots, c_{iN}) + \sum_j E_j(c_{1j}, \ldots, c_{Nj}) \tag{7}$$

According to Equation 3, the computation of a single factor-to-variable message involves maximizing over 2^n configurations. E and I, however, are binary constraints and evaluate to $-\infty$ for most configurations. This drastically reduces the number of configurations which can maximize the message values. Given this simple fact, Givoni and Frey (2009) show how to reduce the necessary update messages to only two types of scalar ones: *availabilities* (α) and *responsibilities* (ρ).[2]

A responsibility message ρ_{ij}, sent from a variable node c_{ij} to function node E_j, reflects the evidence of how likely j is to be an exemplar for i given all other potential exemplars:

$$\rho_{ij} = \mathcal{SIM}(i,j) - \max_{k \neq j}(\mathcal{SIM}(i,k) + \alpha_{ik}) \tag{8}$$

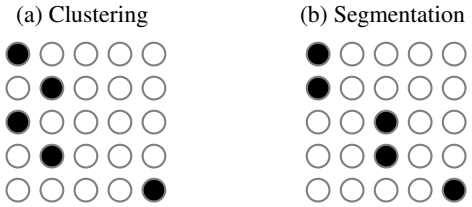
An availability message α_{ij}, sent from a function node E_j to a variable node c_{ij}, reflects how likely point j is to be an exemplar for i given the evidence from all other data points:

$$\alpha_{ij} = \begin{cases} \sum_{k \neq j} \max[\rho_{kj}, 0] & \text{if } i = j \\ \min[0, \rho_{jj} + \sum_{k \notin \{i,j\}} \max[\rho_{kj}, 0]] & \text{if } i \neq j \end{cases} \tag{9}$$

Let $\gamma_{ij}(l)$ be the message value corresponding to setting variable c_{ij} to l, $l \in \{0, 1\}$. Instead of sending two-valued messages (corresponding to the two possible values of the binary variables), we can send the difference for the two possible configurations: $\gamma_{ij} = \gamma_{ij}(1) - \gamma_{ij}(0)$ – effectively, a log-likelihood ratio.

[2]Normally, each iteration of the algorithm sends five types of two-valued messages: to and from functions E and I and a message from functions S. Fortunately, the messages sent to and from E factors to the variable nodes subsume the three other message types and it is not necessary to compute them explicitly. See (Givoni and Frey, 2009, p.195) for details.

Figure 3: Examples of valid configuration of hidden variables $\{c_{ij}\}$ for clustering and segmentation.



The algorithm converges when the set of points labelled as exemplars remains unchanged for a predetermined number of iterations. When the algorithm terminates, messages to each variable are added together. A positive final message indicates that the most likely value of a variable c_{ij} is 1 (point j is an exemplar for i), a negative message indicates that it is 0 (j is not an exemplar for i).

4 Affinity Propagation for Segmentation

This section explains how we adapt the Affinity Propagation clustering algorithm to segmentation.

In this setting, sentences are data points and we refer to exemplars as *segment centres*. Given a document, we want to assign each sentence to a segment centre so as to maximize net similarity.

The new formulation relies on the same underlying factor graph (Figure 2). A binary variable node c_{ij} is set to 1 iff sentence j is the segment centre for sentence i. When clustering is the objective, a cluster may consist of points coming from anywhere in the data sequence. When segmentation is the objective, a segment must consist of a solid block of points around the segment centre. Figure 3 shows, for a toy problem with 5 data points, possible valid configurations of variables $\{c_{ij}\}$ for clustering (3a) and for segmentation (3b).

To formalize this new linearity requirement, we elaborate Equation 4 into Equation 10. E_j evaluates to $-\infty$ in three cases. Case 1 is the original coherence constraint. Case 2 states that no point k may be in the segment with a centre is j, if k lies before the start of the segment (the sequence $c_{(s-1)j} = 0$, $c_{sj} = 1$ necessarily corresponds to the start of the segment). Case 3 handles analogously the end of the segment.

$$E_j = \begin{cases} -\infty & \textbf{1}.\ \text{if } c_{jj} = 0 \ \wedge\ c_{ij} = 1 \text{ for some } i \neq j \\ & \textbf{2}.\ \text{if } c_{jj} = 1 \ \wedge\ c_{sj} = 1 \ \wedge\ c_{(s-1)j} = 0 \\ & \wedge\, c_{kj} = 1 \text{ for some } s < j,\ k < s-1 \\ & \textbf{3}.\ \text{if } c_{jj} = 1 \ \wedge\ c_{ej} = 1 \ \wedge\ c_{(e+1)j} = 0 \\ & \wedge\, c_{kj} = 1 \text{ for some } e > j,\ k > e+1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The *E* function nodes are the only changed part of the factor graph, so we only must re-derive α messages (availabilities) sent from factors *E* to variable nodes. A function-to-variable message is computed as shown in Equation 11 (elaborated Equation 3), and the only incoming messages to E nodes are responsibilities (ρ messages):

$$\mu_{f\to x} = \max_{N(f)\setminus x}(f(x_1,\ldots,x_m) + \sum_{x'\in N(f)\setminus x} \mu_{x'\to f}) = \tag{11}$$

$$\max_{c_{ij},\, i\neq j}((E_j(c_{1j},\ldots,c_{Nj}) + \sum_{c_{ij},\, i\neq j} \rho_{ij}(c_{ij})))$$

We need to compute the message values for the two possible settings of binary variables – denoted as $\alpha_{ij}(1)$ and $\alpha_{ij}(0)$ – and propagate the difference $\alpha_{ij} = \alpha_{ij}(1)$ - $\alpha_{ij}(0)$.

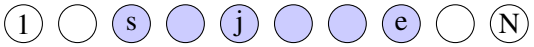
Consider the case of factor E_j sending an α message to the variable node c_{jj} (i.e., $i = j$). If $c_{jj} = 0$ then point j is not its own segment centre and the only valid configuration is to set all other c_{ij} to 0:

$$\alpha_{jj}(0) = \max_{c_{ij}, i\neq j}(E_j(c_{1j},\ldots,c_{Nj}) + \sum_{c_{ij}, i\neq j} \rho_{ij}(c_{ij})) \tag{12}$$

$$= \sum_{i\neq j} \rho_{ij}(0)$$

To compute $\alpha_{ij}(1)$ (point j is its own segment centre), we only must maximize over configurations which will not correspond to cases 2 and 3 in Equation 10 (other assignments are trivially non-optimal because they would evaluate E_j to $-\infty$). Let the start of a segment be s, $1 \leq s < j$ and the end of the segment be e, $j+1 < e \leq N$. We only need to consider configurations such that all points between s and e are in the segment while all others are not.

The following picture shows a valid configuration.[3]



To compute the message $\alpha_{ij}(1), i = j$, we have:

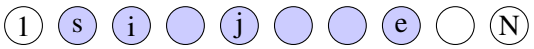
$$\alpha_{jj}(1) = \max_{s=1}^{j}[\sum_{k=1}^{s-1}\rho_{kj}(0) + \sum_{k=s}^{j-1}\rho_{kj}(1)] + \quad (13)$$
$$\max_{e=j}^{N}[\sum_{k=j+1}^{e}\rho_{kj}(1) + \sum_{k=e+1}^{N}\rho_{kj}(0)]$$

Subtracting Equation 12 from Equation 13, we get:

$$\alpha_{jj} = \alpha_{jj}(1) - \alpha_{jj}(0) = \quad (14)$$
$$\max_{s=1}^{j}(\sum_{k=s}^{j-1}\rho_{kj}) + \max_{e=j}^{N}(\sum_{k=j+1}^{e}\rho_{kj})$$

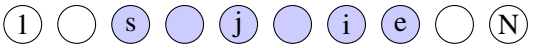
Now, consider the case of factor E_j sending an α message to a variable node c_{ij} other than segment exemplar j (i.e., $i \neq j$). Two subcases are possible: point i may lie before the segment centre j ($i < j$), or it may lie after the segment centre ($i > j$).

The configurations which may maximize $\alpha_{ij}(1)$ (the message value for setting the hidden variable to 1) necessarily conform to two conditions: point j is labelled as a segment centre (c_{jj} = 1) and all points lying between i and j are in the segment. This corresponds to Equation 15 for $i < j$ and to Equation 16 for $i > j$. Pictorial examples of corresponding valid configurations precede the equations.



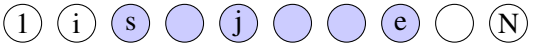
$$\alpha_{ij,\, i<j}(1) = \max_{s=1}^{i}[\sum_{k=1}^{s-1}\rho_{kj}(0) + \sum_{k=s}^{i-1}\rho_{kj}(1)] + \quad (15)$$
$$\sum_{k=i+1}^{j}\rho_{kj}(1) + \max_{e=j}^{N}[\sum_{k=j+1}^{e}\rho_{kj}(1) + \sum_{k=e+1}^{N}\rho_{kj}(0)]$$

[3]Variables c_{ij} set to 1 are shown as shaded circles, to 0 – as white circles. Normally, variables form a column in the factor graph; we transpose them to save space.

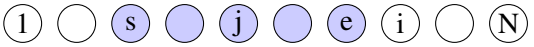


$$\alpha_{ij,\, i>j}(1) = \max_{s=1}^{j}[\sum_{k=1}^{s-1}\rho_{kj}(0) + \sum_{k=s}^{j-1}\rho_{kj}(1)] + \quad (16)$$
$$\sum_{k=j}^{i-1}\rho_{kj}(1) + \max_{e=i}^{N}[\sum_{k=i+1}^{e}\rho_{kj}(1) + \sum_{k=e+1}^{N}\rho_{kj}(0)]$$

To compute the message value for setting the hidden variable c_{ij} to 0, we again distinguish between $i < j$ and $i > j$ and consider whether c_{jj} = 1 or c_{jj} = 0 (point j is / is not a segment centre). For c_{jj} = 0 the only optimal configuration is c_{ij} = 0 for all $i \neq j$. For c_{jj} = 1 the set of possible optimal configurations is determined by the position of point i with respect to point j. Following the same logic as in the previous cases we get Equation 17 for $i < j$ and Equation 18 for $i > j$.



$$\alpha_{ij}(0) = \max(\sum_{k\notin i,j}\rho_{kj}(0), \quad (17)$$
$$\sum_{k=1}^{i-1}\rho_{kj}(0) + \max_{s=i+1}^{j}[\sum_{k=i+1}^{s-1}\rho_{kj}(0) + \sum_{k=s}^{j-1}\rho_{kj}(1)] +$$
$$\rho_{jj}(1) + \max_{e=j}^{N}[\sum_{k=j+1}^{e}\rho_{kj}(1) + \sum_{k=e+1}^{N}\rho_{kj}(0)])$$



$$\alpha_{ij}(0) = \max(\sum_{k\notin i,j}\rho_{kj}(0), \quad (18)$$
$$\max_{s=1}^{j}[\sum_{k=1}^{s-1}\rho_{kj}(0) + \sum_{k=s}^{j-1}\rho_{kj}(1)] +$$
$$\rho_{jj}(1) + \max_{e=j}^{i-1}[\sum_{k=j+1}^{e}\rho_{kj}(1) + \sum_{k=e+1}^{i-1}\rho_{kj}(0)]$$
$$\sum_{k=i+1}^{N}\rho_{kj}(0))$$

Due to space constraints, we will omit the details of subtracting Equation 17 from 15 and Equation 18 from 16. The final update rules for both $i < j$ and

Algorithm 1 Affinity Propagation for Segmentation

1: **input**: 1) a set of pairwise similarities $\{\mathcal{SIM}(i,j)\}_{(i,j)\in\{1,\ldots,N\}^2}$, $\mathcal{SIM}(i,j) \in \mathbb{R}$; 2) a set of preferences (self-similarities) $\{\mathcal{SIM}(i,i)\}_{i\in\{1,\ldots,N\}}$ indicating *a priori* likelihood of point i being a segment centre

2: **initialization**: $\forall i,j : \alpha_{ij} = 0$ (set all availabilities to 0)

3: **repeat**

4: iteratively update responsibilities (ρ) and availabilities (α)

5:
$$\forall i,j : \rho_{ij} = \mathcal{SIM}(i,j) + \max_{k\neq j}(\mathcal{SIM}(i,k) - \alpha_{ik})$$

6:
$$\forall i,j : \alpha_{ij} = \begin{cases} \max_{s=1}^{j}(\sum_{k=s}^{j-1}\rho_{kj}) + \max_{e=j}^{N}(\sum_{k=j+1}^{e}\rho_{kj}) & \text{if } i=j \\ \min[\max_{s=1}^{i}\sum_{k=s}^{i-1}\rho_{kj} + \sum_{k=i+1}^{j}\rho_{kj} + \max_{e=j}^{N}\sum_{k=j+1}^{e}\rho_{kj}, & \\ \quad \max_{s=1}^{i}\sum_{k=s}^{i-1}\rho_{kj} + \min_{s=i+1}^{j}\sum_{k=i+1}^{s-1}\rho_{kj}] & \text{if } i<j \\ \min[\max_{s=1}^{j}\sum_{k=s}^{j-1}\rho_{kj} + \sum_{k=j}^{i-1}\rho_{kj} + \max_{e=i}^{N}\sum_{k=i+1}^{e}\rho_{kj}, & \\ \quad \min_{e=j}^{i-1}\sum_{k=e+1}^{i-1}\rho_{kj} + \max_{e=i}^{N}\sum_{k=i+1}^{e}\rho_{kj}] & \text{if } i>j \end{cases}$$

7: **until** convergence

8: compute the final configuration of variables: $\forall i, j$ j is the exemplar for i iff $\rho_{ij} + \alpha_{ij} > 0$

9: **output**: exemplar assignments

$i > j$ appear in Algorithm 1, where we summarize the whole process.

The equations look cumbersome but they are trivial to compute. Every summand corresponds to finding the most likely start or end of the segment, taking into account *fixed* information. When computing messages for any given sender node, we can remember the maximizing values for neighbouring recipient nodes. For example, after computing the availability message from factor E_j to c_{ij}, we must only consider one more responsibility value when computing the message from E_j to variable $c_{(i+1)j}$. The cost of computing a message is thus negligible.

When the matrix is fully specified, each iteration requires passing $2N^2$ messages, so the algorithm runs in $O(N^2)$ time and requires $O(N^2)$ memory (to store the similarities, the availabilities and the responsibilities). When performing segmentation, however, the user generally has some idea about the average or maximum segment length. In such more realistic cases, the input matrix of similarities is sparse – it is constructed by sliding a window of size M. M usually needs to be at least twice the maximum segment length or thrice the average segment length. Each iteration, then, involves sending $2MN$ messages and the storage requirements are also $O(MN)$.

As is common in loopy belief propagation algorithms, both availability and responsibility messages are dampened to avoid overshooting and oscillating. The dampening factor is λ where $0.5 \leq \lambda < 1$.

$$newMsg = \lambda * oldMsg + (1-\lambda)newMsg \quad (19)$$

The *APS* algorithm is unsupervised. It only benefits

from a small development set to fine-tune a few parameters: preference values and the dampening factor. *APS* does not require (nor allow) specifying the number of segments beforehand. The granularity of segmentation is adjusted through preference values; this reflect how likely each sentence is to be selected as a segment centre. (This translates into the cost of adding a segment.)

Because each message only requires the knowledge about one column or row of the matrix, the algorithm can be easily parallelized.

5 Experimental Setting

Datasets. We evaluate the performance of the *APS* algorithm on three datasets. The first, compiled by Malioutov and Barzilay (2006), consists of manually transcribed and segmented lectures on Artificial Intelligence, 3 development files and 19 test files. The second dataset consists of 227 chapters from medical textbooks (Eisenstein and Barzilay, 2008), 5 of which we use for development. In this dataset the gold standard segment boundaries correspond to section breaks specified by the authors. The third dataset consists of 85 works of fiction downloaded from Project Gutenberg, 3 of which are used for development. The segment boundaries correspond to chapter breaks or to breaks between individual stories. They were inserted automatically using HTML markup in the downloaded files.

The datasets exhibit different characteristics. The lecture dataset and the fiction dataset are challenging because they are less cohesive than medical textbooks. The textbooks are cognitively more difficult to process and the authors rely on repetition of terminology to facilitate comprehension. Since lexical repetition is the main source of information for text segmentation, we expect a higher performance on this dataset. Transcribed speech, on the other hand, is considerably less cohesive. The lecturer makes an effort to speak in "plain language" and to be comprehensible, relying less on terminology. The use of pronouns is very common, as is the use of examples.

Repeated use of the same words is also uncommon in fiction. In addition, the dataset was compiled automatically using HTML markup. The markup is not always reliable and occasionally the e-book proofreaders skip it altogether, which potentially adds noise to the dataset.

Baselines. We compare the performance of *APS* with that of two state-of-the-art segmenters: the Minimum Cut segmenter (Malioutov and Barzilay, 2006) and the Bayesian segmenter (Eisenstein and Barzilay, 2008). The authors have made Java implementations publicly available. For the Minimum Cut segmenter, we select the best parameters using the script included with that distribution. The Bayesian segmenter automatically estimates all necessary parameters from the data.

Preprocessing and the choice of similarity metric. As described in Section 4, the *APS* algorithm takes as inputs a matrix of pairwise similarities between sentences in the document and also, for each sentence, a preference value.

This paper focuses on comparing globally informed segmentation algorithms, and leaves for future work the exploration of best similarity metrics. To allow fair comparison, then, we use the same metric as the Minimum Cut segmenter, cosine similarity. Each sentence is represented as a vector of token-type frequencies. Following (Malioutov and Barzilay, 2006), the frequency vectors are smoothed by adding counts of words from the adjacent sentences and then weighted using a tf.idf metric (for details, see *ibid.*) The similarity between sentence vectors s_1 and s_2 is computed as follows:

$$cos(s_1, s_2) = \frac{s_1 \bullet s_2}{||s_1|| \times ||s_2||} \tag{20}$$

The representation used by the Bayesian segmenter is too different to be incorporated into our model directly, but ultimately it is based on the distribution of unigrams in documents. This is close enough to our representation to allow fair comparison.

The fiction dataset consists of books: novels or collections of short stories. Fiction is known to exhibit less lexical cohesion. That is why – when working on this dataset – we work at the paragraph level: the similarity is measured not between sentences but between paragraphs. We use this representation with all three segmenters.

All parameters have been fine-tuned on the development portions of the datasets. For *APS* algorithm *per se* we needed to set three parameters: the size of the sliding window for similarity computations, the dampening factor λ and the preference values. The

	BayesSeg	MinCutSeg	*APS*
AI	0.443	0.437	**0.404**
Clinical	**0.353**	0.382	0.371
Fiction	0.377	0.381	**0.350**

Table 1: Results of segmenting the three datasets using the Bayesian segmenter, the Minimum Cut segmenter and *APS*.

parameters for the similarity metric (best variation of tf.idf, the window size and the decay factor for smoothing) were set using the script provided in the Minimum Cut segmenter's distribution.

Evaluation metric. We have measured the performance of the segmenters with the WindowDiff metric (Pevzner and Hearst, 2002). It is computed by sliding a window through reference and through segmentation output and, at each window position, comparing the number of reference breaks to the number of breaks inserted by the segmenter (hypothetical breaks). It is a penalty measure which reports the number of windows where the reference and hypothetical breaks do not match, normalized by the total number of windows. In Equation 21, ref and hyp denote the number of reference and hypothetical segment breaks within a window.

$$winDiff = \frac{1}{N-k} \sum_{i=1}^{N-k} (|ref - hyp| \neq 0) \quad (21)$$

6 Experimental Results and Discussion

Table 1 compares the performance of the three segmenters using WindowDiff values. On the lecture and fiction datasets, the *APS* segmenter outperforms the others by a small margin, around 8% over the better of the two. It is second-best on the clinical textbook dataset. According to a one-tailed paired t-test with 95% confidence cut-off, the improvement is statistically significant only on the fiction dataset. All datasets are challenging and the baselines are very competitive, so drawing definitive conclusions is difficult. Still, we can be fairly confident that *APS* performs at least as well as the other two segmenters. It also has certain advantages.

One important difference between *APS* and the other segmenters is that *APS* does not require the number of segments as an input parameter. This is very helpful, because such information is generally unavailable in any realistic deployment setting. The parameters are fine-tuned to maximize WindowDiff values, so this results in high-precision, low-recall segment assignments; that is because WindowDiff favours missing boundaries over near-hits.

APS also outputs segment centres, thus providing some information about a segment's topic. We have not evaluated how descriptive the segment centres are; this is left for future work.

APS performs slightly better than the other segmenters but not by much. We hypothesize that one of the reasons is that *APS* relies on the presence of descriptive segment centres which are not necessarily present for large, coarse-grained segments such as chapters in novels. It is possible for *APS* to have an advantage performing fine-grained segmentation.

7 Conclusions and Future Work

In this paper we have presented *APS* – a new algorithm for linear text segmentation. *APS* takes into account the global structure of the document and outputs segment boundaries and segment centres. It scales linearly in the number of input sentences, performs competitively with the state-of-the-art and is easy to implement. We also provide a Java implementation of the *APS* segmenter.

We consider two main directions for future work: using more informative similarity metrics and making the process of segmentation hierarchical. We chose to use cosine similarity primarily to allow fair comparison and to judge the algorithm itself, in isolation from the information it uses. Cosine similarity is a very simple metric which cannot provide an adequate picture of topic fluctuations in documents. It is likely that dictionary-based or corpus-based similarity measures would yield a major improvement in performance.

Reliance on descriptive segment centres may handicap *APS*'s performance when looking for coarse-grained segments. One possible remedy is to look for shorter segments first and then merge them. One can also modify the algorithm to perform hierarchical segmentation: consider net similarity with low-level segment centres as well as with high-level ones. We plan to explore both possibilities.

Acknowledgements

We thank Inmar Givoni for explaining the details of binary Affinity Propagation and for commenting on our early ideas in this project. Many thanks to Yongyi Mao for a helpful discussion on the use Affinity Propagation for text segmentation.

References

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Blei, David and Pedro Moreno. 2001. Topic Segmentation with an Aspect Hidden Markov Model. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 343–348. ACM Press.

Blei, David M., Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Choi, Freddy Y. Y. 2000. Advances in Domain Independent Linear Text Segmentation. In *Proceedings of NAACL*, pages 26–33.

Eisenstein, Jacob and Regina Barzilay. 2008. Bayesian Unsupervised Topic Segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 334–343, Honolulu, Hawaii, October.

Frey, Brendan J. and Delbert Dueck. 2007. Clustering by Passing Messages Between Data Points. *Science*, 315:972–976.

Givoni, Inmar E. and Brendan J. Frey. 2009. A Binary Variable Model for Affinity Propagation. *Neural Computation*, 21:1589–1600.

Haghighi, Aria and Lucy Vanderwende. 2009. Exploring Content Models for Multi-Document Summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado, June.

Hearst, Marti A. 1997. TextTiling: segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23:33–64, March.

Kschischang, Frank R., Brendan J. Frey, and Hans-A Loeliger. 2001. Factor graphs and the sum-product algorithm. In *IEEE Transactions on Information Theory, Vol 47, No 2*, pages 498–519, February.

Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-01*, pages 282–289.

Malioutov, Igor and Regina Barzilay. 2006. Minimum Cut Model for Spoken Lecture Segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, Sydney, Australia, July.

Misra, Hemant, François Yvon, Joemon M. Jose, and Olivier Cappé. 2009. Text segmentation via topic modeling: an analytical study. In *18th ACM Conference on Information and Knowledge Management*, pages 1553–1556.

Oh, Hyo-Jung, Sung Hyon Myaeng, and Myung-Gil Jang. 2007. Semantic passage segmentation based on sentence topics for question answering. *Information Sciences, an International Journal*, 177:3696–3717, September.

Pearl, Judea. 1982. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, pages 133–136, Pittsburgh, PA.

Pevzner, Lev and Marti A. Hearst. 2002. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 28(1):19–36.

Stoyanov, Veselin and Claire Cardie. 2008. Topic identification for fine-grained opinion analysis. In *COLING '08 Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, pages 817–824.

Youmans, Gilbert. 1991. A new tool for discourse analysis: The vocabulary-management profile. *Language*, 67(4):763–789.

Minimally Supervised Event Causality Identification

Quang Xuan Do **Yee Seng Chan** **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{quangdo2,chanys,danr}@illinois.edu

Abstract

This paper develops a minimally supervised approach, based on focused distributional similarity methods and discourse connectives, for identifying of causality relations between events in context. While it has been shown that distributional similarity can help identifying causality, we observe that discourse connectives and the particular discourse relation they evoke in context provide additional information towards determining causality between events. We show that combining discourse relation predictions and distributional similarity methods in a global inference procedure provides additional improvements towards determining event causality.

1 Introduction

An important part of text understanding arises from understanding the semantics of events described in the narrative, such as identifying the events that are mentioned and how they are related semantically. For instance, when given a sentence "The police arrested him because he killed someone.", humans understand that there are two events, triggered by the words "arrested" and "killed", and that there is a causality relationship between these two events. Besides being an important component of discourse understanding, automatically identifying causal relations between events is important for various natural language processing (NLP) applications such as question answering, etc. In this work, we automatically detect and extract causal relations between events in text.

Despite its importance, prior work on event causality extraction in context in the NLP literature is relatively sparse. In (Girju, 2003), the author used noun-verb-noun lexico-syntactic patterns to learn that "mosquitoes cause malaria", where the *cause* and *effect* mentions are nominals and not necessarily event evoking words. In (Sun et al., 2007), the authors focused on detecting causality between search query pairs in temporal query logs. (Beamer and Girju, 2009) tried to detect causal relations between verbs in a corpus of screen plays, but limited themselves to consecutive, or adjacent verb pairs. In (Riaz and Girju, 2010), the authors first cluster sentences into topic-specific scenarios, and then focus on building a dataset of causal text spans, where each span is headed by a verb. Thus, their focus was not on identifying causal relations between events in a given text document.

In this paper, given a text document, we first identify events and their associated arguments. We then identify causality or relatedness relations between event pairs. To do this, we develop a minimally supervised approach using focused distributional similarity methods, such as co-occurrence counts of events collected automatically from an unannotated corpus, to measure and predict existence of causality relations between event pairs. Then, we build on the observation that discourse connectives and the particular discourse relation they evoke in context provide additional information towards determining causality between events. For instance, in the example sentence provided at the beginning of this section, the words "arrested" and "killed" probably have a relatively high apriori likelihood of being ca-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 294–303,
Edinburgh, Scotland, UK, July 27–31, 2011.

sually related. However, knowing that the connective "because" evokes a contingency discourse relation between the text spans "The police arrested him" and "he killed someone" provides further evidence towards predicting causality. The contributions of this paper are summarized below:

- Our focus is on identifying causality between event pairs in context. Since events are often triggered by either verbs (e.g. "attack") or nouns (e.g. "explosion"), we allow for detection of causality between verb-verb, verb-noun, and noun-noun triggered event pairs. To the best of our knowledge, this formulation of the task is novel.
- We developed a minimally supervised approach for the task using focused distributional similarity methods that are automatically collected from an unannotated corpus. We show that our approach achieves better performance than two approaches: one based on a frequently used metric that measures association, and another based on the effect-control-dependency (ECD) metric described in a prior work (Riaz and Girju, 2010).
- We leverage on the interactions between event causality prediction and discourse relations prediction. We combine these knowledge sources through a global inference procedure, which we formalize via an Integer Linear Programming (ILP) framework as a constraint optimization problem (Roth and Yih, 2004). This allows us to easily define appropriate constraints to ensure that the causality and discourse predictions are coherent with each other, thereby improving the performance of causality identification.

2 Event Causality

In this work, we define an event as an action or occurrence that happens with associated participants or arguments. Formally, we define an event e as: $p(a_1, a_2, \ldots, a_n)$, where the predicate p is the word that triggers the presence of e in text, and $a_1, a_2, \ldots, a_n$ are the arguments associated with e. Examples of predicates could be *verbs* such as "attacked", "employs", *nouns* such as "explosion", "protest", etc., and examples of the arguments of "attacked" could be its *subject* and *object* nouns.

To measure the causality association between a pair of events e_i and e_j (in general, e_i and e_j could be extracted from the same or different documents), we should use information gathered about their predicates and arguments. A simple approach would be to directly calculate the pointwise mutual information (PMI)[1] between $p^i(a_1^i, a_2^i, \ldots, a_n^i)$ and $p^j(a_1^j, a_2^j, \ldots, a_m^j)$. However, this leads to very sparse counts as the predicate p^i with its list of arguments $a_1^i, \ldots, a_n^i$ would rarely co-occur (within some reasonable context distance) with predicate p^j and its entire list of arguments $a_1^j, \ldots, a_m^j$. Hence, in this work, we measure causality association using three separate components and focused distributional similarity methods collected about event pairs as described in the rest of this section.

2.1 Cause-Effect Association

We measure the causality or cause-effect association (CEA) between two events e_i and e_j using the following equation:

$$CEA(e_i, e_j) = s_{pp}(e_i, e_j) + s_{pa}(e_i, e_j) + s_{aa}(e_i, e_j) \quad (1)$$

where s_{pp} measures the association between event predicates, s_{pa} measures the association between the predicate of an event and the arguments of the other event, and s_{aa} measures the association between event arguments. In our work, we regard each event e as being triggered and rooted at a predicate p.

2.1.1 Predicate-Predicate Association

We define s_{pp} as follows:

$$s_{pp}(e_i, e_j) = PMI(p^i, p^j) \times \max(u^i, u^j) \times IDF(p^i, p^j) \times Dist(p^i, p^j) \quad (2)$$

which takes into account the PMI between predicates p^i and p^j of events e_i and e_j respectively, as well as various other pieces of information. In Suppes' *Probabilistic theory of Casuality* (Suppes, 1970), he highlighted that event e is a possible cause of event e', if e' happens more frequently with e than

[1] PMI is frequently used to measure association between variables.

by itself, i.e. $P(e'|e) > P(e')$. This can be easily rewritten as $\frac{P(e,e')}{P(e)P(e')} > 1$, similar to the definition of PMI:

$$PMI(e, e') = log\frac{P(e, e')}{P(e)P(e')}$$

which is only positive when $\frac{P(e,e')}{P(e)P(e')} > 1$.

Next, we build on the intuition that event predicates appearing in a large number of documents are probably not important or discriminative. Thus, we penalize these predicates when calculating s_{pp} by adopting the inverse document frequency (idf):

$$IDF(p^i, p^j) = idf(p^i) \times idf(p^j) \times idf(p^i, p^j),$$

where $idf(p) = log\frac{D}{1+N}$, D is the total number of documents in the collection and N is the number of documents that p occurs in.

We also award event pairs that are closer together, while penalizing event pairs that are further apart in texts, by incorporating the distance measure of Leacock and Chodorow (1998), which was originally used to measure similarity between concepts:

$$Dist(p^i, p^j) = -log\frac{|sent(p^i) - sent(p^j)| + 1}{2 \times ws},$$

where $sent(p)$ gives the sentence number (index) in which p occurs and ws indicates the window-size (of sentences) used. If p^i and p^j are drawn from the same sentence, the numerator of the above fraction will return 1. In our work, we set ws to 3 and thus, if p^i occurs in sentence k, the furthest sentence that p^j will be drawn from, is sentence $k + 2$.

The final component of Equation 2, $\max(u^i, u^j)$, takes into account whether predicates (events) p^i and p^j appear most frequently with each other. u^i and u^j are defined as follows:

$$u^i = \frac{P(p^i, p^j)}{\max_k[P(p^i, p^k)] - P(p^i, p^j) + \epsilon}$$
$$u^j = \frac{P(p^i, p^j)}{\max_k[P(p^k, p^j)] - P(p^i, p^j) + \epsilon},$$

where we set $\epsilon = 0.01$ to avoid zeros in the denominators. u^i will be maximized if there is no other predicate p^k having a higher co-occurrence probability with p^i, i.e. $p^k = p^j$. u^j is treated similarly.

2.1.2 Predicate-Argument and Argument-Argument Association

We define s_{pa} as follows:

$$s_{pa}(e_i, e_j) = \frac{1}{|A_{e_j}|}\sum_{a \in A_{e_j}} PMI(p^i, a) + \frac{1}{|A_{e_i}|}\sum_{a \in A_{e_i}} PMI(p^j, a), \quad (3)$$

where A_{e_i} and A_{e_j} are the sets of arguments of e_i and e_j respectively.

Finally, we define s_{aa} as follows:

$$s_{aa}(e_i, e_j) = \frac{1}{|A_{e_i}||A_{e_j}|}\sum_{a \in A_{e_i}}\sum_{a' \in A_{e_j}} PMI(a, a') \quad (4)$$

Together, s_{pa} and s_{aa} provide additional contexts and robustness (in addition to s_{pp}) for measuring the cause-effect association between events e_i and e_j.

Our formulation of CEA is inspired by the ECD metric defined in (Riaz and Girju, 2010):

$$ECD(a, b) = \max(v, w) \times -log\frac{dis(a, b)}{2 \times maxDistance}, \quad (5)$$

where

$$v = \frac{P(a, b)}{P(b) - P(a, b) + \epsilon} \times \frac{P(a, b)}{\max_t[P(a, b_t)] - P(a, b) + \epsilon}$$
$$w = \frac{P(a, b)}{P(a) - P(a, b) + \epsilon} \times \frac{P(a, b)}{\max_t[P(a_t, b)] - P(a, b) + \epsilon},$$

where ECD(a,b) measures the causality between two events a and b (headed by verbs), and the second component in the ECD equation is similar to $Dist(p^i, p^j)$. In our experiments, we will evaluate the performance of ECD against our proposed approach.

So far, our definitions in this section are generic and allow for any list of event argument types. In this work, we focus on two argument types: agent (subject) and patient (object), which are typical core arguments of any event. We describe how we extract event predicates and their associated arguments in the section below.

3 Verbal and Nominal Predicates

We consider that events are not only triggered by verbs but also by nouns. For a verb (verbal predicate), we extract its subject and object from its associated dependency parse. On the other hand, since

events are also frequently triggered by nominal predicates, it is important to identify an appropriate list of event triggering nouns. In our work, we gathered such a list using the following approach:

- We first gather a list of deverbal nouns from the set of most frequently occurring (in the Gigaword corpus) 3,000 verbal predicate types. For each verb type v, we go through all its WordNet[2] senses and gather all its derivationally related nouns $\mathcal{N}_v$ [3].

- From $\mathcal{N}_v$, we heuristically remove nouns that are less than three characters in length. We also remove nouns whose first three characters are different from the first three characters of v. For each of the remaining nouns in $\mathcal{N}_v$, we measured its Levenstein (edit) distance from v and keep the noun(s) with the minimum distance. When multiple nouns have the same minimum distance from v, we keep all of them.

- To further prune the list of nouns, we next removed all nouns ending in "er", "or", or "ee", as these nouns typically refer to a person, e.g. "writer", "doctor", "employee". We also remove nouns that are not hyponyms (children) of the first WordNet sense of the noun "event"[4].

- Since we are concerned with nouns denoting events, FrameNet (Ruppenhofer et al., 2010) (FN) is a good resource for mining such nouns. FN consists of frames denoting situations and events. As part of the FN resource, each FN frame consists of a list of lexical units (mainly verbs and nouns) representing the semantics of the frame. Various frame-to-frame relations are also defined (in particular the *inheritance* relation). Hence, we gathered all the children frames of the FN frame "Event". From these children frames, we then gathered all their noun lexical units (words) and add them to our list of nouns. Finally, we also add a few nouns denoting natural disaster from Wikipedia[5].

Using the above approach, we gathered a list of about 2,000 noun types. This current approach is heuristics based which we intend to improve in the future, and any such improvements should subsequently improve the performance of our causality identification approach.

Event triggering deverbal nouns could have associated arguments (for instance, acting as subject, object of the deverbal noun). To extract these arguments, we followed the approach of (Gurevich et al., 2008). Briefly, the approach uses linguistic patterns to extract subjects and objects for deverbal nouns, using information from dependency parses. For more details, we refer the reader to (Gurevich et al., 2008).

4 Discourse and Causality

Discourse connectives are important for relating different text spans, helping us to understand a piece of text in relation to its context:

[The police arrested him] <u>because</u> [he killed someone].

In the example sentence above, the discourse connective ("because") and the discourse relation it evokes (in this case, the *Cause* relation) allows readers to relate its two associated text spans, "The police arrested him" and "he killed someone". Also, notice that the verbs "arrested" and "killed", which *cross* the two text spans, are causally related. To aid in extracting causal relations, we leverage on the identification of discourse relations to provide additional contextual information.

To identify discourse relations, we use the Penn Discourse Treebank (PDTB) (Prasad et al., 2007), which contains annotations of discourse relations in context. The annotations are done over the Wall Street Journal corpus and the PDTB adopts a predicate-argument view of discourse relations. A discourse connective (e.g. because) takes two text spans as its arguments. In the rest of this section, we briefly describe the discourse relations in PDTB and highlight how we might leverage them to aid in determining event causality.

[2] http://wordnet.princeton.edu/

[3] The WordNet resource provides derivational information on words that are in different syntactic (i.e. part-of-speech) categories, but having the same root (lemma) form and that are semantically related.

[4] The first WordNet sense of the noun "event" has the meaning: "something that happens at a given place and time"

[5] http://en.wikipedia.org/wiki/Natural_disaster

Coarse-grained relations	Fine-grained relations
Comparison	Concession, Contrast, Pragmatic-concession, Pragmatic-contrast
Contingency	Cause, Condition, Pragmatic-cause, Pragmatic-condition
Expansion	Alternative, Conjunction, Exception, Instantiation, List, Restatement
Temporal	Asynchronous, Synchronous

Table 1: Coarse-grained and fine-grained discourse relations.

4.1 Discourse Relations

PDTB contains annotations for four coarse-grained discourse relation types, as shown in the left column of Table 1. Each of these are further refined into several fine-grained discourse relations, as shown in the right column of the table.[6] Next, we briefly describe these relations, highlighting those that could potentially help to determine event causality.

Comparison A *Comparison* discourse relation between two text spans highlights prominent differences between the situations described in the text spans. An example sentence is:

Contrast: [According to the survey, *x*% of Chinese Internet users prefer Google] whereas [*y*% prefer Baidu].

According to the PDTB annotation manual (Prasad et al., 2007), the truth of both spans is independent of the established discourse relation. This means that the text spans are not causally related and thus, the existence of a *Comparison* relation should imply that there is no causality relation across the two text spans.

Contingency A *Contingency* relation between two text spans indicates that the situation described in one text span causally influences the situation in the other. An example sentence is:

Cause: [The first priority is search and rescue] because [many people are trapped under the rubble].

Existence of a *Contingency* relation potentially implies that there exists at least one causal event pair crossing the two text spans. The PDTB annotation manual states that while the *Cause* and *Condition* discourse relations indicate casual influence in their text spans, there is no causal influence in the text spans of the *Pragmatic-cause* and *Pragmatic-condition* relations. For instance, *Pragmatic-condition* indicates that one span provides the context in which the description of the situation in the other span is relevant; for example:

Pragmatic-condition: If [you are thirsty], [there's beer in the fridge].

Hence, there is a need to also identify fine-grained discourse relations.

Expansion Connectives evoking *Expansion* discourse relations expand the discourse, such as by providing additional information, illustrating alternative situations, etc. An example sentence is:

Conjunction: [Over the past decade, *x* women were killed] and [*y* went missing].

Most of the *Expansion* fine-grained relations (except for *Conjunction*, which could connect arbitrary pieces of text spans) should not contain causality relations across its text spans.

Temporal These indicate that the situations described in the text spans are related temporally. An example sentence is:

Synchrony: [He was sitting at his home] when [the whole world started to shake].

Temporal precedence of the (cause) event over the (effect) event is a necessary, but not sufficient requisite for causality. Hence by itself, *Temporal* relations are probably not discriminative enough for determining event causality.

4.2 Discourse Relation Extraction System

Our work follows the approach and features described in the state-of-the-art Ruby-based discourse system of (Lin et al., 2010), to build an in-house Java-based discourse relation extraction system. Our system identifies explicit connectives in text, predict their discourse relations, as well as their associated text spans. Similar to (Lin et al., 2010), we achieved a competitive performance of slightly over 80% F1-score in identifying fine-grained relations for explicit connectives. Our system is developed using the Learning Based Java modeling lan-

[6] PDTB further refines these fine-grained relations into a final third level of relations, but we do not use them in this work.

guage (LBJ) (Rizzolo and Roth, 2010) and will be made available soon. Due to space constraints, we refer interested readers to (Lin et al., 2010) for details on the features, etc.

In the example sentences given thus far in this section, all the connectives were explicit, as they appear in the texts. PDTB also provides annotations for implicit connectives, which we do not use in this work. Identifying implicit connectives is a harder task and incorporating these is a possible future work.

5 Joint Inference for Causality Extraction

To exploit the interactions between event pair causality extraction and discourse relation identification, we define appropriate constraints between them, which can be enforced through the Constrained Conditional Models framework (aka ILP for NLP) (Roth and Yih, 2007; Chang et al., 2008). In doing this, the predictions of CEA (Section 2.1) and the discourse system are forced to cohere with each other. More importantly, this should improve the performance of using only CEA to extract causal event pairs. To the best of our knowledge, this approach for causality extraction is novel.

5.1 CEA & Discourse: Implementation Details

Let $\mathcal{E}$ denote the set of event mentions in a document. Let $\mathcal{EP} = \{(e_i, e_j) \in \mathcal{E} \times \mathcal{E} \mid e_i \in \mathcal{E}, e_j \in \mathcal{E}, i < j, |sent(e_i) - sent(e_j)| \leq 2\}$ denote the set of event mention pairs in the document, where $sent(e)$ gives the sentence number in which event e occurs. Note that in this work, we only extract event pairs that are at most two sentences apart. Next, we define $\mathcal{L}_{ER}$ = {"causal", "¬ causal"} to be the set of event relation labels that an event pair $ep \in \mathcal{EP}$ can be associated with.

Note that the CEA metric as defined in Section 2.1 simply gives a score without it being bounded to be between 0 and 1.0. However, to use the CEA score as part of the inference process, we require that it be bounded and thus can be used as a binary prediction, that is, predicting an event pair as *causal* or ¬*causal*. To enable this, we use a few development documents to automatically find a threshold CEA score that separates scores indicating *causal* vs ¬*causal*. Based on this threshold, the original CEA scores are then rescaled to fall within 0 to 1.0. More details on this are in Section 6.2.

Let $\mathcal{C}$ denote the set of connective mentions in a document. We slightly modify our discourse system as follows. We define $\mathcal{L}_{DR}$ to be the set of discourse relations. We initially add all the fine-grained discourse relations listed in Table 1 to $\mathcal{L}_{DR}$. In the PDTB corpus, some connective examples are labeled with just a coarse-grained relation, without further specifying a fine-grained relation. To accommodate these examples, we add the coarse-grained relations *Comparison*, *Expansion*, and *Temporal* to $\mathcal{L}_{DR}$. We omit the coarse-grained *Contingency* relation from $\mathcal{L}_{DR}$, as we want to separate *Cause* and *Condition* from *Pragmatic-cause* and *Pragmatic-condition*. This discards very few examples as only a very small number of connective examples are simply labeled with a *Contingency* label without further specifying a fine-grained label. We then retrained our discourse system to predict labels in $\mathcal{L}_{DR}$.

5.2 Constraints

We now describe the constraints used to support joint inference, based on the predictions of the CEA metric and the discourse classifier. Let $s_c(dr)$ be the probability that connective c is predicated to be of discourse relation dr, based on the output of our discourse classifier. Let $s_{ep}(er)$ be the CEA prediction score (rescaled to range in [0,1]) that event pair ep takes on the *causal* or ¬*causal* label er. Let $x_{\langle c,dr\rangle}$ be a binary indicator variable which takes on the value 1 iff c is labeled with the discourse relation dr. Similarly, let $y_{\langle ep,er\rangle}$ be a binary variable which takes on the value 1 iff ep is labeled as er. We then define our objective function as follows:

$$\max \Big[|\mathcal{L}_{DR}| \sum_{c \in \mathcal{C}} \sum_{dr \in \mathcal{L}_{DR}} s_c(dr) \cdot x_{\langle c,dr\rangle} + |\mathcal{L}_{ER}| \sum_{ep \in \mathcal{EP}} \sum_{er \in \mathcal{L}_{ER}} s_{ep}(er) \cdot y_{\langle ep,er\rangle} \Big] \quad (6)$$

subject to the following constraints:

$$\sum_{dr \in \mathcal{L}_{DR}} x_{\langle c,dr\rangle} = 1 \quad \forall c \in \mathcal{C} \quad (7)$$

$$\sum_{er \in \mathcal{L}_{ER}} y_{\langle ep,er\rangle} = 1 \quad \forall ep \in \mathcal{EP} \quad (8)$$

$$x_{\langle c,dr\rangle} \in \{0,1\} \quad \forall c \in \mathcal{C}, dr \in \mathcal{L}_{DR} \quad (9)$$

$$y_{\langle ep,er\rangle} \in \{0,1\} \quad \forall ep \in \mathcal{EP}, er \in \mathcal{L}_{ER} \quad (10)$$

Equation (7) requires that each connective c can only be assigned one discourse relation. Equation (8) requires that each event pair ep can only be *causal* or $\neg$*causal*. Equations (9) and (10) indicate that $x_{\langle c,dr\rangle}$ and $y_{\langle ep,er\rangle}$ are binary variables.

To capture the relationship between event pair causality and discourse relations, we use the following constraints:

$$x_{\langle c,\text{“}Cause\text{”}\rangle} \leq \sum_{ep\in\mathcal{EP}_c} y_{\langle ep,\text{“}causal\text{”}\rangle} \quad (11)$$

$$x_{\langle c,\text{“}Condition\text{”}\rangle} \leq \sum_{ep\in\mathcal{EP}_c} y_{\langle ep,\text{“}causal\text{”}\rangle}, \quad (12)$$

where both equations are defined $\forall c \in \mathcal{C}$. $\mathcal{EP}_c$ is defined to be the set of event pairs that cross the two text spans associated with c. For instance, if the first text span of c contains two event mentions e_i, e_j, and there is one event mention e_k in the second text span of c, then $\mathcal{EP}_c = \{(e_i, e_k), (e_j, e_k)\}$. Finally, the logical form of Equation (11) can be written as: $x_{\langle c,\text{“}Cause\text{”}\rangle} \Rightarrow y_{\langle ep_i,\text{“}causal\text{”}\rangle} \vee \ldots \vee y_{\langle ep_j,\text{“}causal\text{”}\rangle}$, where $ep_i, \ldots, ep_j$ are elements in $\mathcal{EP}_c$. This states that if we assign the *Cause* discourse label to c, then at least one of $ep_i, \ldots, ep_j$ must be assigned as *causal*. The interpretation of Equation (12) is similar.

We use two more constraints to capture the interactions between event causality and discourse relations. First, we defined $\mathcal{C}_{ep}$ as the set of connectives c enclosing each event of ep in each of its text spans, i.e.: one of the text spans of c contain one of the event in ep, while the other text span of c contain the other event in ep. Next, based on the discourse relations in Section 4.1, we propose that when an event pair ep is judged to be *causal*, then the connective c that encloses it should be evoking one of the discourse relations in $\mathcal{L}_{DR_a}$ = {“Cause”, “Condition”, “Temporal”, “Asynchronous”, “Synchrony”, “Conjunction”}. We capture this using the following constraint:

$$y_{\langle ep,\text{“}causal\text{”}\rangle} \leq \sum_{dr_a\in\mathcal{L}_{DR_a}} x_{\langle c,dr_a\rangle} \quad \forall c \in \mathcal{C}_{ep} \quad (13)$$

The logical form of Equation (13) can be written as: $y_{\langle ep,\text{“}causal\text{”}\rangle} \Rightarrow x_{\langle c,\text{“}Cause\text{”}\rangle} \vee x_{\langle c,\text{“}Condition\text{”}\rangle} \ldots \vee x_{\langle c,\text{“}Conjunction\text{”}\rangle}$. This states that if we assign ep as *causal*, then we must assign to c one of the labels in $\mathcal{L}_{DR_a}$.

Finally, we propose that for any connectives evoking discourse relations $\mathcal{L}_{DR_b}$ = {“Comparison”, “Concession”, “Contrast”, “Pragmatic-concession”, “Pragmatic-contrast”, “Expansion”, “Alternative”, “Exception”, “Instantiation”, “List”, “Restatement”}, any event pair(s) that it encloses should be $\neg$*causal*. We capture this using the following constraint:

$$x_{\langle c,dr_b\rangle} \leq y_{\langle ep,\text{“}\neg causal\text{”}\rangle} \quad \forall\ dr_b \in \mathcal{L}_{DR_b},\ ep \in \mathcal{EP}_c, \quad (14)$$

where the logical form of Equation (14) can be written as: $x_{\langle c,dr_b\rangle} \Rightarrow y_{\langle ep,\text{“}\neg causal\text{”}\rangle}$.

6 Experiments

6.1 Experimental Settings

To collect the distributional statistics for measuring CEA as defined in Equation (1), we applied part-of-speech tagging, lemmatization, and dependency parsing (Marneffe et al., 2006) on about 760K documents in the English Gigaword corpus (LDC catalog number LDC2003T05).

We are not aware of any benchmark corpus for evaluating event causality extraction in contexts. Hence, we created an evaluation corpus using the following process: Using news articles collected from CNN[7] during the first three months of 2010, we randomly selected 20 articles (documents) as evaluation data, and 5 documents as development data.

Two annotators annotated the documents for causal event pairs, using two simple notions for causality: the Cause event should temporally precede the Effect event, and the Effect event occurs because the Cause event occurs. However, sometimes it is debatable whether two events are involved in a causal relation, or whether they are simply involved in an uninteresting temporal relation. Hence, we allowed annotations of C to indicate causality, and R to indicate relatedness (for situations when the existence of causality is debatable). The annotators will simply identify and annotate the *C* or *R* relations between predicates of event pairs. Event arguments are not explicitly annotated, although the annotators are free to look at the entire document text while making their annotation decisions. Finally, they are free

[7] http://www.cnn.com

System	Rec%	Pre%	F1%
PMI_{pp}	26.6	20.8	23.3
ECD_{pp} &$PMI_{pa,aa}$	40.9	23.5	29.9
CEA	62.2	28.0	38.6
CEA+Discourse	65.1	30.7	41.7

Table 2: Performance of baseline systems and our approaches on extracting *Causal* event relations.

System	Rec%	Pre%	F1%
PMI_{pp}	27.8	24.9	26.2
ECD_{pp} &$PMI_{pa,aa}$	42.4	28.5	34.1
CEA	63.1	33.7	43.9
CEA+Discourse	65.3	36.5	46.9

Table 3: Performance of the systems on extracting *Causal* and *Related* event relations.

to annotate relations between predicates that have any number of sentences in between and are not restricted to a fixed sentence window-size.

After adjudication, we obtained a total of 492 $C+R$ relation annotations, and 414 C relation annotations on the evaluation documents. On the development documents, we obtained 92 $C+R$ and 71 C relation annotations. The annotators overlapped on 10 evaluation documents. On these documents, the first (second) annotator annotated 215 (199) $C+R$ relations, agreeing on 166 of these relations. Together, they annotated 248 distinct relations. Using this number, their agreement ratio would be 0.67 $(166/248)$. The corresponding agreement ratio for C relations is 0.58. These numbers highlight that causality identification is a difficult task, as there could be as many as N^2 event pairs in a document (N is the number of events in the document). We plan to make this annotated dataset available soon.[8]

6.2 Evaluation

As mentioned in Section 5.1, to enable translating (the unbounded) CEA scores into binary *causal*, ¬*causal* predictions, we need to rescale or calibrate these scores to range in [0,1]. To do this, we first rank all the CEA scores of all event pairs in the development documents. Most of these event pairs will be ¬*causal*. Based on the relation annotations in these development documents, we scanned through

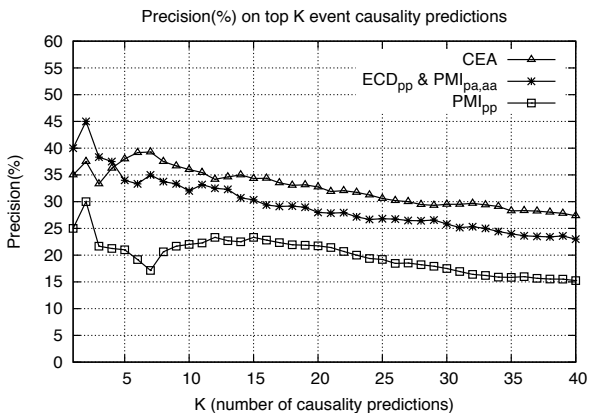


Figure 1: Precision of the top K causality C predictions.

this ranked list of scores to locate the CEA score *t* that gives the highest F1-score (on the development documents) when used as a threshold between *causal* vs ¬*causal* decisions. We then ranked all the CEA scores of all event pairs gathered from the 760K Gigaword documents, discretized all scores higher than *t* into B bins, and all scores lower than t into B bins. Together, these $2B$ bins represent the range [0,1]. We used $B = 500$. Thus, consecutive bins represent a difference of 0.001 in calibrated scores.

To measure the causality between a pair of events e_i and e_j, a simple baseline is to calculate $PMI(p^i, p^j)$. Using a similar thresholding and calibration process to translate $PMI(p^i, p^j)$ scores into binary causality decisions, we obtained a F1 score of 23.1 when measured over the causality C relations, as shown in the row PMI_{pp} of Table 2.

As mentioned in Section 2.1.2, Riaz and Girju (2010) proposed the ECD metric to measure causality between two events. Thus, as a point of comparison, we replaced s_{pp} of Equation (1) with $ECD(a, b)$ of Equation (5), substituting $a = p^i$ and $b = p^j$. After thresholding and calibrating the scores of this approach, we obtained a F1-score of 29.7, as shown in the row ECD_{pp}&$PMI_{pa,aa}$ of Table 2.

Next, we evaluated our proposed CEA approach and obtained a F1-score of 38.6, as shown in the row CEA of Table 2. Thus, our proposed approach obtained significantly better performance than the PMI baseline and the ECD approach. Next, we performed joint inference with the discourse relation predictions as described in Section 5 and obtained

[8] http://cogcomp.cs.illinois.edu/page/publication_view/663

an improved F1-score of 41.7. We note that we obtained improvements in both recall and precision. This means that with the aid of discourse relations, we are able to recover more causal relations, as well as reduce false-positive predictions.

Constraint Equations (11) and (12) help to recover causal relations. For improvements in precision, as stated in the last paragraph of Section 5.2, identifying other discourse relations such as "Comparison", "Contrast", etc., provides counter-evidence to causality. Together with constraint Equation (14), this helps to eliminate false-positive event pairs as classified by CEA and contributes towards CEA+Discourse having a higher precision than CEA.

The corresponding results for extracting both causality and relatedness $C + R$ relations are given in Table 3. For these experiments, the aim was for a more relaxed evaluation and we simply collapsed C and R into a single label.

Finally, we also measured the precision of the top K causality C predictions, showing the precision trends in Figure 1. As shown, CEA in general achieves higher precision when compared to PMI_{pp} and $ECD_{pp}\&PMI_{pa,aa}$. The trends for $C + R$ predictions are similar.

Thus far, we had included both verbal and nominal predicates in our evaluation. When we repeat the experiments for $ECD_{pp}\&PMI_{pa,aa}$ and CEA on just verbal predicates, we obtained the respective F1-scores of 31.8 and 38.3 on causality relations. The corresponding F1-scores for casuality and relatedness relations are 35.7 and 43.3. These absolute F1-scores are similar to those in Tables 2 and 3, differing by 1-2%.

7 Analysis

We randomly selected 50 false-positive predictions and 50 false-negative *causality* relations to analyze the mistakes made by CEA.

Among the false-positives (precision errors), the most frequent error type (56% of the errors) is that CEA simply assigns a high score to event pairs that are not causal; more knowledge sources are required to support better predictions in these cases. The next largest group of error (22%) involves events containing pronouns (e.g. "he", "it") as arguments. Applying coreference to replace these pronouns with their canonical entity strings or labeling them with semantic class information might be useful.

Among the false-negatives (recall errors), 23% of the errors are due to CEA simply assigning a low score to causal event pairs and more contextual knowledge seems necessary for better predictions. 19% of the recall errors arises from causal event pairs involving nominal predicates that are not in our list of event evoking noun types (described in Section 3). A related 17% of recall errors involves nominal predicates without any argument. For these, less information is available for CEA to make predictions. The remaining group (15% of errors) involves events containing pronouns as arguments.

8 Related Work

Although prior work in event causality extraction in context is relatively sparse, there are many prior works concerning other semantic aspects of event extraction. Ji and Grishman (2008) extracts event mentions (belonging to a predefined list of target event types) and their associated arguments. In other prior work (Chen et al., 2009; Bejan and Harabagiu, 2010), the authors focused on identifying another type of event pair semantic relation: event coreference. Chambers and Jurafsky (2008; 2009) chain events sharing a common (protagonist) participant. They defined events as verbs and given an existing chain of events, they predict the next likely event involving the protagonist. This is different from our task of detecting causality between arbitrary event pairs that might or might not share common arguments. Also, we defined events more broadly, as those that are triggered by either verbs or nouns. Finally, although our proposed CEA metric has resemblance the ECD metric in (Riaz and Girju, 2010), our task is different from theirs and our work differs in many aspects. They focused on building a dataset of causal text spans, whereas we focused on identifying causal relations between events in a given text document. They considered text spans headed by verbs while we considered events triggered by both verbs and nouns. Moreover, we combined event causality prediction and discourse relation prediction through a global inference procedure to further improve the performance of event causality prediction.

9 Conclusion

In this paper, using general tools such as the dependency and discourse parsers which are not trained specifically towards our target task, and a minimal set of development documents for threshold tuning, we developed a minimally supervised approach to identify causality relations between events in context. We also showed how to incorporate discourse relation predictions to aid event causality predictions through a global inference procedure. There are several interesting directions for future work, including the incorporation of other knowledge sources such as coreference and semantic class predictions, which were shown to be potentially important in our error analysis. We could also use discourse relations to aid in extracting other semantic relations between events.

Acknowledgments

The authors thank the anonymous reviewers for their insightful comments and suggestions. University of Illinois at Urbana-Champaign gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract No. FA8750-09-C-0181. The first author thanks the Vietnam Education Foundation (VEF) for its sponsorship. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the VEF, DARPA, AFRL, or the US government.

References

Brandon Beamer and Roxana Girju. 2009. Using a bigram event model to predict causal potential. In *CICLING*.

Cosmin Adrian Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *ACL*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL-HLT*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *ACL*.

Ming-Wei Chang, Lev Ratinov, Nicholas Rizzolo, and Dan Roth. 2008. Learning and inference with constraints. In *AAAI*.

Zheng Chen, Heng Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *RANLP workshop on Events in Emerging Text Types*.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *ACL workshop on Multilingual Summarization and Question Answering*.

Olga Gurevich, Richard Crouch, Tracy Holloway King, and Valeria de Paiva. 2008. Deverbal nouns in knowledge representation. *Journal of Logic and Computation*, 18, June.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through unsupervised cross-document inference. In *ACL*.

Claudia Leacock and Martin Chodorow, 1998. *Combining Local Context and WordNet Similarity for Word Sense Identification*. MIT Press.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2010. A pdtb-styled end-to-end discourse parser. Technical report. http://www.comp.nus.edu.sg/ linzihen/publications/tech2010.pdf.

Marie-catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie Webber. 2007. The penn discourse treebank 2.0 annotation manual. Technical report. http://www.seas.upenn.edu/ pdtb/PDTBAPI/pdtb-annotation-manual.pdf.

Mehwish Riaz and Roxana Girju. 2010. Another look at causality: Discovering scenario-specific contingency relationships with no supervision. In *ICSC*.

N. Rizzolo and D. Roth. 2010. Learning based java for rapid development of nlp systems. In *LREC*.

Dan Roth and Wen Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *CoNLL*.

Dan Roth and Wen Tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2010. *FrameNet II: Extended Theory and Practice*. http://framenet.icsi.berkeley.edu.

Yizhou Sun, Ning Liu, Kunqing Xie, Shuicheng Yan, Benyu Zhang, and Zheng Chen. 2007. Causal relation of queries from temporal logs. In *WWW*.

Patrick Suppes. 1970. *A Probabilistic Theory of Causality*. Amsterdam: North-Holland Publishing Company.

A Model of Discourse Predictions in Human Sentence Processing

Amit Dubey and **Frank Keller** and **Patrick Sturt**
Human Communication Research Centre, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
{amit.dubey,frank.keller,patrick.sturt}@ed.ac.uk

Abstract

This paper introduces a psycholinguistic model of sentence processing which combines a Hidden Markov Model noun phrase chunker with a co-reference classifier. Both models are fully incremental and generative, giving probabilities of lexical elements conditional upon linguistic structure. This allows us to compute the information theoretic measure of surprisal, which is known to correlate with human processing effort. We evaluate our surprisal predictions on the Dundee corpus of eye-movement data show that our model achieve a better fit with human reading times than a syntax-only model which does not have access to co-reference information.

1 Introduction

Recent research in psycholinguistics has seen a growing interest in the role of *prediction* in sentence processing. Prediction refers to the fact that the human sentence processor is able to anticipate upcoming material, and that processing is facilitated when predictions turn out to be correct (evidenced, e.g., by shorter reading times on the predicted word or phrase). Prediction is presumably one of the factors that contribute to the efficiency of human language understanding. Sentence processing is *incremental* (i.e., it proceeds on a word-by-word basis); therefore, it is beneficial if unseen input can be anticipated and relevant syntactic and semantic structure constructed in advance. This allows the processor to save time and makes it easier to cope with the constant stream of new input.

Evidence for prediction has been found in a range of psycholinguistic processing domains. *Semantic prediction* has been demonstrated by studies that show anticipation based on selectional restrictions: listeners are able to launch eye-movements to the predicted argument of a verb before having encountered it, e.g., they will fixate an edible object as soon as they hear the word *eat* (Altmann and Kamide, 1999). Semantic prediction has also been shown in the context of semantic priming: a word that is preceded by a semantically related prime or by a semantically congruous sentence fragment is processed faster (Stanovich and West, 1981; Clifton et al., 2007). An example for *syntactic prediction* can be found in coordinate structures: readers predict that the second conjunct in a coordination will have the same syntactic structure as the first conjunct (Frazier et al., 2000). In a similar vein, having encountered the word *either*, readers predict that *or* and a conjunct will follow it (Staub and Clifton, 2006). Again, priming studies corroborate this: Comprehenders are faster at naming words that are syntactically compatible with prior context, even when they bear no semantic relationship to it (Wright and Garrett, 1984).

Predictive processing is not confined to the sentence level. Recent experimental results also provide evidence for *discourse prediction*. An example is the study by van Berkum et al. (2005), who used a context that made a target noun highly predictable, and found a mismatch effect in the ERP (event-related brain potential) when an adjective appeared that was inconsistent with the target noun. An example is (we give translations of their Dutch materials):

(1) The burglar had no trouble locating the secret family safe.
   a. Of course, it was situated behind a

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 304–312,
Edinburgh, Scotland, UK, July 27–31, 2011.

big$_{neu}$ but unobtrusive painting$_{neu}$.

b. Of course, it was situated behind a big$_{com}$ but unobtrusive bookcase$_{com}$.

Here, the adjective *big*, which can have neutral or common gender in Dutch, is consistent with the predicted noun *painting* in (1-a), but inconsistent with it in (1-b), leading to a mismatch ERP on *big* in (1-b) but not in (1-a).

Previous results on discourse effects in sentence processing can also be interpreted in terms of prediction. In a classical paper, Altmann and Steedman (1988) demonstrated that PP-attachment preferences can change through discourse context: if the context contains two potential referents for the target NP, then NP-attachment of a subsequent PP is preferred (to disambiguate between the two referents), while if the context only contains one target NP, VP-attachment is preferred (as there is no need to disambiguate). This result (and a large body of related findings) is compatible with an interpretation in which the processor predicts upcoming syntactic attachment based on the presence of referents in the preceding discourse.

Most attempts to model prediction in human language processing have focused on syntactic prediction. Examples include Hale's (2001) surprisal model, which relates processing effort to the conditional probability of the current word given the previous words in the sentence. This approach has been elaborated by Demberg and Keller (2009) in a model that explicitly constructs predicted structure, and includes a verification process that incurs additional processing cost if predictions are not met. Recent work has attempted to integrate semantic and discourse prediction with models of syntactic processing. This includes Mitchell et al.'s (2010) approach, which combines an incremental parser with a vector-space model of semantics. However, this approach only provides a loose integration of the two components (through simple addition of their probabilities), and the notion of semantics used is restricted to lexical meaning approximated by word co-occurrences. At the discourse level, Dubey (2010) has proposed a model that combines an incremental parser with a probabilistic logic-based model of co-reference resolution. However, this model does not explicitly model discourse effects in terms of prediction, and again only proposes a loose integration of co-reference and syntax. Furthermore, Dubey's (2010) model has only been tested on two experimental data sets (pertaining to the interaction of ambiguity resolution with context), no broad coverage evaluation is available.

The aim of the present paper is to overcome these limitations. We propose a computational model that captures discourse effects on syntax in terms of prediction. The model comprises a co-reference component which explicitly stores discourse mentions of NPs, and a syntactic component which adjust the probabilities of NPs in the syntactic structure based on the mentions tracked by the discourse component. Our model is HMM-based, which makes it possible to efficiently process large amounts of data, allowing an evaluation on eye-tracking corpora, which has recently become the gold-standard in computational psycholinguistics (e.g., Demberg and Keller 2008; Frank 2009; Boston et al. 2008; Mitchell et al. 2010).

The paper is structured as follows: In Section 2, we describe the co-reference and the syntactic models and evaluate their performance on standard data sets. Section 3 presents an evaluation of the overall model on the Dundee eye-tracking corpus. The paper closes with a comparison with related work and a general discussion in Sections 4 and 5.

2 Model

This model utilises an NP chunker based upon a hidden Markov model (HMM) as an approximation to syntax. Using a simple model such as an HMM facilitates the integration of a co-reference component, and the fact that the model is generative is a prerequisite to using surprisal as our metric of interest (as surprisal require the computation of prefix probabilities). The key insight in our model is that human sentence processing is, on average, facilitated when a previously-mentioned discourse entity is repeated. This facilitation depends upon keeping track of a list of previously-mentioned entities, which requires (at the least) shallow syntactic information, yet the facilitation itself is modeled primarily as a lexical phenomenon. This allows a straightforward separation of concerns: shallow syntax is captured using the HMM's hidden states, whereas the co-reference fa-

cilitation is modeled using the HMM's emissions. The vocabulary of hidden states is described in Section 2.1 and the emission distribution in Section 2.2

2.1 Syntactic Model

A key feature of the co-reference component of our model (described below) is that syntactic analysis and co-reference resolution happen simultaneously. This could potentially slow down the syntactic analysis, which tends to already be quite slow for exhaustive surprisal-based incremental parsers. Therefore, rather than using full parsing, we use an HMM-based NP chunker which allows for a fast analysis. NP chunking is sufficient to extract NP discourse mentions and, as we show below, surprisal values computed using HMM chunks provide a useful fit on the Dundee eye-movement data.

To allow the HMM to handle possessive constructions as well as NP with simple modifiers and complements, the HMM decodes NP subtrees with depth of 2, by encoding the start, middle and end of a syntactic category X as '(X', 'X' and 'X)', respectively. To reduce an explosion in the number of states, the category begin state '(X' only appears at the rightmost lexical token of the constituent's leftmost daughter. Likewise, 'X)' only appears at the leftmost lexical token of the constituent's rightmost daughter. An example use of this state vocabulary can be seen in Figure 1. Here, a small degree of recursion allows for the NP ((new york city's) general obligation fund) to be encoded, with the outer NP's left bracket being 'announced' at the token *'s*, which is the rightmost lexical token of the inner NP. Hidden states also include part-of-speech (POS) tags, allowing simultaneous POS tagging. In the example given in Figure 1, the full state can be read by listing the labels written above a word, from top to bottom. For example, the full state associated with *'s* is (NP-NP)-POS. As *'s* can also be a contraction of *is*, another possible state for *'s* is VBZ (without recursive categories as we are only interested in NP chunks).

The model uses unsmoothed bi-gram transition probabilities, along with a maximum entropy distribution to guess unknown word features. The resulting distribution has the form $P(tag|word)$ and is therefore unsuitable for computing surprisal values. However, using Bayes' theorem we can compute:

$$P(word|tag) = \frac{P(tag|word)P(word)}{P(tag)} \quad (1)$$

which is what we need for surprisal. The primary information from this probability comes from $P(tag|word)$, however, reasonable estimates of $P(tag)$ and $P(word)$ are required to ensure the probability distribution is proper. $P(tag)$ may be estimated on a parsed treebank. $P(word)$, the probability of a particular unseen word, is difficult to estimate directly. Given that our training data contains approximately 10^6 words, we assume that this probability must be bounded above by 10^{-6}. As an approximation, we use this upper bound as the probability of $P(word)$.

Training The chunker is trained on sections 2–22 of the Wall Street Journal section of the Penn Treebank. CoNLL 2000 included chunking as a shared task, and the results are summarized by Tjong Kim Sang and Buchholz (2000). Our chunker is not comparable to the systems in the shared task for several reasons: we use more training data, we tag simultaneously (the CoNLL systems used gold standard tags) and our notion of a chunk is somewhat more complex than that used in CoNLL. The best performing chunker from CoNLL 2000 achieved an F-score of 93.5%, and the worst performing system an F-score of 85.8%. Our chunker achieves a comparable F-score of 85.5%, despite the fact that it simultaneously tags and chunks, and only uses a bigram model.

2.2 Co-Reference Model

In a standard HMM, the emission probabilities are computed as $P(w_i|s_i)$ where w_i is the i^{th} word and s_i is the i^{th} state. In our model, we replace this with a choice between two alternatives:

$$P(w_i|s_i) = \begin{cases} \lambda P_{\text{seen before}}(w_i|s_i) \\ (1-\lambda) P_{\text{discourse new}}(w_i|s_i) \end{cases} \quad (2)$$

The 'discourse new' probability distribution is the standard HMM emission distribution. The 'seen before' distribution is more complicated. It is in part based upon caching language models. However, the contents of the cache are not individual words but

						(NP	NP	NP	NP)					
(NP	NP)		(NP	NP	NP	NP)	NP					(NP	NP	NP)
JJ	NN	IN	NNP	NNP	NNP	POS	JJ	NN	NNS	VBN	RP	DT	NN	NN
strong	demand	for	new	york	city	's	general	obligation	bonds	propped	up	the	municipal	market

Figure 1: The chunk notation of a tree from the training data.

Variable	Type
l, l'	List of trie nodes
w, w_i	Words
t	Tag
n, n'	Trie nodes

$l \leftarrow List(\text{root of mention trie})$
for $w \leftarrow w_0$ to w_n **do**
  $l' \leftarrow l$
  $l \leftarrow \emptyset$
  Clear tag freq array ft
  Clear word freq array fwt
  for $t \in$ tag set **do**
    for $n \in l'$ **do**
      $ft(t) \leftarrow ft(t) + FreqOf(n,t)$
      $n' \leftarrow Getchild(w,t)$
      if $n' \neq \emptyset$ **then**
        $fwt(t) \leftarrow fwt(t) + FreqOf(n', w, t)$
        $l \leftarrow n' :: l$
      end if
    end for
  end for
  $P_{\text{seen before}}(w|t) = ft(t)/fwt(t)$
end for

Figure 2: Looking up entries from the NP Cache

rather a collection of all NPs mentioned so far in the document.

Using a collection of NPs rather than individual words complicates the decoding process. If m is the size of a document, and n is the size of the current sentence, decoding occurs in $O(mn)$ time as opposed to $O(n)$, as the collection of NPs needs to be accessed at each word. However, we do not store the NPs in a list, but rather a trie. This allows decoding to occur in $O(n \log m)$ time, which we have found to be quite fast in practise. The algorithm used to keep track of currently active NPs is presented in Figure 2. This shows how the distribution $P_{\text{seen before}}$ is updated on a word-by-word basis. At the end of each sentence, the NPs of the Viterbi parse are added to the mention trie after having their leading articles stripped. A weakness of the algorithm is that mentions are only added on a sentence-by-sentence basis (disallowing within-sentence references). Although the algorithm is intended to find whole-string matches, in practise, it will count any NP whose prefix matches as being co-referent.

A consequence of Equation 2 is that co-reference resolution is handled at the same time as HMM decoding. Whenever the 'seen before' distribution is applied, an NP is co-referent with one occurring earlier. Likewise, whenever the 'discourse new' distribution is applied, the NP is not co-referent with any NP appearing previously. As one choice or the other is made during decoding, the decoder therefore also selects a chain of co-referent entities. Generally, for words which *have* been used in this discourse, the magnitude of probabilities in the 'seen before' distribution are much higher than in the 'discourse new' distribution. Thus, there is a strong bias to classify NPs which match word-for-word as being co-referent. There remains a possibility that the model primarily captures lexical priming, rather than co-reference. However, we note that string match is a strong indicator of two NPs being corefer-

ent (cf. Soon et al. 2001), and, moreover, the matching is done on an NP-by-NP basis, which is more suitable for finding entity coreference, rather than a word-by-word basis, which would be more suitable for lexical priming.

An appealing side-effect of using a simple co-reference decision rule which is applied incrementally is that it is relatively simple to incrementally compute the transitive closure of co-reference chains, resulting in the entity sets which are then used in evaluation.

The co-reference model only has one free parameter, λ, which is estimated from the ACE-2 corpus. The estimate is computed by counting how often a repeated NP actually is discourse new. In the current implementation of the model, λ is constant throughout the test runs. However, λ could possibly be a function of the previous discourse, allowing for more complicated classification probabilities.

3 Evaluation

3.1 Data

Our evaluation experiments were conducted upon the Dundee corpus (Kennedy et al., 2003), which contains the eye-movement record of 10 participants each reading 2,368 sentences of newspaper text. This data set has previously been used by Demberg and Keller (2008) and Frank (2009) among others.

3.2 Evaluation

Eye tracking data is noisy for a number of reasons, including the fact that experimental participants can look at any word which is currently displayed. While English is normally read in a left-to-right manner, readers often skip words or make regressions (i.e., look at a word to the left of the one they are currently fixating). Deviations from a strict left-to-right progression of fixations motivate the need for several different measures of eye movement. The model presented here predicts the Total Time that participants spent looking at a region, which includes any re-fixations after looking away. In addition to total time, other possible measures include (a) First Pass, which measures the initial fixation and any re-fixations before looking at any other word (this occurs, for instance, if the eye initially lands at the start of a long word – the eye will tend to re-fixate on a more central viewing location), (b) Right Bounded reading time, which includes all fixations on a word before moving to the right of the word (i.e., re-fixations after moving left are included), and (c) Second Pass, which includes any re-fixation on a word after looking at any other word (be it to the left or the right of the word of interest). We found that the model performed similarly across all these reading time metrics, we therefore only report results for Total Time.

As mentioned above, reading measures are hypothesised to correlate with Surprisal, which is defined as:

$$S(w_t) = -\log(P(w_t|w_1...w_{t1}) \quad (3)$$

We compute the surprisal scores for the syntax-only HMM, which does not have access to co-reference information (henceforth referred to as 'HMM') and the full model, which combines the syntax-only HMM with the co-reference model (henceforth 'HMM+Ref'). To determine if our Dundee corpus simulations provide a reasonable model of human sentence processing, we perform a regression analysis with the Dundee corpus reading time measure as the dependent variable and the surprisal scores as the independent variable.

To account for noise in the corpus, we also use a number of additional explanatory variables which are known to strongly influence reading times. These include the logarithm of the frequency of a word (measured in occurrences per million) and the length of a word in letters. Two additional explanatory variables were available in the Dundee corpus, which we also included in the regression model. These were the position of a word on a line, and which line in a document a word appeared in. As participants could only view one line at a time (i.e., one line per screen), these covariates are known as line position and screen position, respectively.

All the covariates, including the surprisal estimates, were centered before including them in the regression model. Because the HMM and HMM+Ref surprisal values are highly collinear, the HMM+Ref surprisal values were added as residuals of the HMM surprisal values.

In a normal regression analysis, one must either assume that participants or the particular choice of

items add some randomness to the experiment, and either each participant's responses for all items must be averaged (treating participants as a random factor), or all participant's responses for each item is averaged (treating items as a random factor). However, in the present analysis we utilise a mixed effects model, which allows both items and participants to be treated as random factors.[1]

The are a number of criteria which can be used to test the efficacy of one regression model over another. These include the Aikake Information Criterion (AIC), the Bayesian Information Criterion (BIC), which trade off model fit and number of model parameters (lower scores are better). It is also common to compare the log-likelihood of the models (higher log-likelihood is better), in which case a χ^2 can be used to evaluate if a model offers a significantly better fit, given the number of parameters is uses. We test three models: (i) a baseline, with only low-level factors as independent variables; (ii) the HMM model, with the baseline factors plus surprisal computed by the syntax-only HMM; and (iii) the HMM+Ref model which includes the raw surprisal values of the syntax-only HMM and the surprisal of the HMM+Ref models as computed as a residual of the HMM surprisal score. We compare the HMM and HMM+Ref to the baseline, and the HMM+Ref model against the HMM model.

Some of the data needed to be trimmed. If, due to data sparsity, the surprisal of a word goes to infinity for one of the models, we entirely remove that word from the analysis. This occurred seven times form the HMM+Ref model, but did not occur at all with the HMM model. Some of the eye-movement data was trimmed, as well. Fixations on the first and last words of a line were excluded, as were tracklosses. However, we did not trim any items due to abnormally short or abnormally long fixation durations.

[1] We assume that each participant and item bias the reading time of the experiment. Such an analysis is known as having random intercepts of participant and item. It is also possible to assume a more involved analysis, known as random slopes, where the participants and items bias the slope of the predictor. The model did not converge when using random intercept and slopes on both participant and item. If random slopes on items were left out, the HMM regression model did converge, but not the HMM+Ref model. As the HMM+Ref is the model of interest random slopes were left out entirely to allow a like-with-like comparison between the HMM and HMM+Ref regression models.

3.3 Results

The result of the model comparison on Total Time reading data is summarised in Table 1. To allow this work to be compared with other models, the lower part of the table gives the abosolute AIC, BIC and log likelihood of the baseline model, while the upper part gives delta AIC, BIC and log likelihood scores of pairs of models.

We found that both the HMM and HMM+Ref provide a significantly better fit with the reading time data than the Baseline model; all three criteria agree: AIC and BIC lower than for the baseline, and log-likelihood is higher. Moreover, the HMM+Ref model provides a significantly better fit than the HMM model, which demonstrates the benefit of co-reference information for modeling reading times. Again, all three measures provide the same result.

Table 2 corroborates this result. It list the mixed-model coefficients for the HMM+Ref model and shows that all factors are significant predictors, including both HMM surprisal and residualized HMM+Ref surprisal.

4 Related Work

There have been few computational models of human sentence processing that have incorporated a referential or discourse-level component. Niv (1994) proposed a parsing model based on Combinatory Categorial Grammar (Steedman, 2001), in which referential information was used to resolve syntactic ambiguities. The model was able to capture effects of referential information on syntactic garden paths (Altmann and Steedman, 1988). This model differs from that proposed in the present paper, as it is intended to capture psycholinguistic preferences in a qualitative manner, whereas the aim of the present model is to provide a *quantitative* fit to measures of processing difficulty. Moreover, the model was not based on a large-scale grammar, and was not tested on unrestricted text. Spivey and Tanenhaus (1998) proposed a sentence processing model that examined the effects of referential information, as well as other constraints, on the resolution of ambiguous sentences. Unlike Niv (1994),

From	To	Δ AIC	Δ BIC	Δ logLik	χ^2	Significance
Baseline	HMM	-80	-69	41	82.112	$p < .001$
Baseline	HMM+Ref	-99	-89	51	101.54	$p < .001$
HMM	HMM+Ref	-19	-8	11	21.424	$p < .001$

Model	AIC	BIC	logLik
Baseline	10567789	10567880	-5283886

Table 1: Model comparison (upper part) and absolute scores for the Baseline model (lower part)

Coefficient	Estimate	Std Error	t-value
(Intercept)	991.4346	23.7968	41.66
log(Word Frequency)	-55.3045	1.4830	-37.29
Word Length	128.6216	1.4677	87.63
Screen Position	-1.7769	0.1326	-13.40
Line Position	10.1592	0.7387	13.75
HMM	12.1287	1.3366	9.07
HMM+Ref	19.2772	4.1627	4.63

Table 2: Coefficients of the HMM+Ref model on Total Reading Times. Note that $t > 2$ indicates that the factor in question is a significant predictor.

Spivey and Tanenhaus's (1998) model was specifically designed to provide a quantitative fit to reading times. However, the model lacked generality, being designed to deal with only one type of sentence. In contrast to both of these earlier models, the model proposed here aims to be general enough to provide estimated reading times for unrestricted text. In fact, as far as we are aware, the present paper represents the first wide-coverage model of human parsing that has incorporated discourse-level information.

5 Discussion

The primary finding of this work is that incorporating discourse information such as co-reference into an incremental probabilistic model of sentence processing has a beneficial effect on the ability of the model to predict broad-coverage human parsing behaviour.

Although not thoroughly explored in this paper, our finding is related to an ongoing debate about the structure of the human sentence processor. In particular, the model of Dubey (2010), which also simulates the effect of discourse on syntax, is aimed at examining *interactivity* in the human sentence processor. Interactivity describes the degree to which human parsing is influenced by non-syntactic factors. Under the *weakly interactive* hypothesis, discourse factors may prune or re-weight parses, but only when assuming the *strongly interactive* hypothesis would we argue that the sentence processor predicts upcoming material due to discourse factors. Dubey found that a weakly interactive model simulated a pattern of results in an experiment (Grodner et al., 2005) which was previously believed to provide evidence for the strongly interactive hypothesis. However, as Dubey does not provide broad-coverage parsing results, this leaves open the possibility that the model cannot generalise beyond the experiments expressly modeled in Dubey (2010).

The model presented here, on the other hand, is not only broad-coverage but could also be described as a strongly interactive model. The strong interactivity arises because co-reference resolution is strongly tied to lexical generation probabilities, which are part of the syntactic portion of our model. This cannot be achieve in a weakly interactive model, which is limited to pruning or re-weighting of parses based on discourse information. As our analysis on the Dundee corpus showed, the lexical probabilities (in the form of HMM+Ref surprisal) are key to improving the fit on eye-tracking data. We therefore argue that our results provide evidence

against a weakly interactive approach, which may be sufficient to model individual phenomena (as shown by Dubey 2010), but is unlikely to be able to match the broad-coverage result we have presented here. We also note that psycholinguistic evidence for discourse prediction (such as the context based lexical prediction shown by van Berkum et al. 2005, see Section 1) is also evidence for strong interactivity; prediction goes beyond mere pruning or reweighting and requires strong interactivity.

References

Gerry Altmann and Mark Steedman. Interaction with context during human sentence processing. *Cognition*, 30:191–238, 1988.

Gerry T. M. Altmann and Yuki Kamide. Incremental interpretation at verbs: Restricting the domain of subsequent reference. *Cognition*, 73:247–264, 1999.

Marisa Ferrara Boston, John T. Hale, Reinhold Kliegl, and Shravan Vasisht. Surprising parser actions and reading difficulty. In *Proceedings of ACL-08:HLT, Short Papers*, pages 5–8, 2008.

Charles Clifton, Adrian Staub, and Keith Rayner. Eye movement in reading words and sentences. In R V Gompel, M Fisher, W Murray, and R L Hill, editors, *Eye Movements: A Window in Mind and Brain*, pages 341–372. Elsevier, 2007.

Vera Demberg and Frank Keller. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109: 192–210, 2008.

Vera Demberg and Frank Keller. A computational model of prediction in human parsing: Unifying locality and surprisal effects. In *Proceedings of the 29th meeting of the Cognitive Science Society (CogSci-09)*, 2009.

Amit Dubey. The influence of discourse on syntax: A psycholinguistic model of sentence processing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, Uppsala, Sweden, 2010.

Stefan Frank. Surprisal-based comparison between a symbolic and a connectionist model of sentence processing. In *31st Annual Conference of the Cognitive Science Society (COGSCI 2009)*, Amsterdam, The Netherlands, 2009.

Lyn Frazier, Alan Munn, and Charles Clifton. Processing coordinate structure. *Journal of Psycholinguistic Research*, 29:343–368, 2000.

Daniel J. Grodner, Edward A. F. Gibson, and Duane Watson. The influence of contextual constrast on syntactic processing: Evidence for strong-interaction in sentence comprehension. *Cognition*, 95(3):275–296, 2005.

John T. Hale. A probabilistic earley parser as a psycholinguistic model. In *In Proceedings of the Second Meeting of the North American Chapter of the Asssociation for Computational Linguistics*, 2001.

A. Kennedy, R. Hill, and J. Pynte. The dundee corpus. In *Proceedings of the 12th European conference on eye movement*, 2003.

Jeff Mitchell, Mirella Lapata, Vera Demberg, and Frank Keller. Syntactic and semantic factors in processing difficulty: An integrated measure. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, 2010.

M. Niv. A psycholinguistically motivated parser for CCG. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*, pages 125–132, Las Cruces, NM, 1994.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27 (4):521–544, 2001.

M. J. Spivey and M. K. Tanenhaus. Syntactic ambiguity resolution in discourse: Modeling the effects of referential context and lexical frequency. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 24(6):1521–1543, 1998.

Kieth E. Stanovich and Richard F. West. The effect of sentence context on ongoing word recognition: Tests of a two-pricess theory. *Journal of Experimental Psychology: Human Perception and Performance*, 7:658–672, 1981.

Adrian Staub and Charles Clifton. Syntactic prediction in language comprehension: Evidence from

either . . . or. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 32:425–436, 2006.

Mark Steedman. *The Syntactic Process*. Bradford Books, 2001.

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

Jos J. A. van Berkum, Colin M. Brown, Pienie Zwitserlood, Valesca Kooijman, and Peter Hagoort. Anticipating upcoming words in discourse: Evidence from erps and reading times. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 31(3):443–467, 2005.

Barton Wright and Merrill F. Garrett. Lexical decision in sentences: Effects of syntactic structure. *Memory and Cognition*, 12:31–45, 1984.

Simple Effective Decipherment via Combinatorial Optimization

Taylor Berg-Kirkpatrick and **Dan Klein**
Computer Science Division
University of California at Berkeley
{tberg, klein}@cs.berkeley.edu

Abstract

We present a simple objective function that when optimized yields accurate solutions to both decipherment and cognate pair identification problems. The objective simultaneously scores a matching between two alphabets and a matching between two lexicons, each in a different language. We introduce a simple coordinate descent procedure that efficiently finds effective solutions to the resulting combinatorial optimization problem. Our system requires only a list of words in both languages as input, yet it competes with and surpasses several state-of-the-art systems that are both substantially more complex and make use of more information.

1 Introduction

Decipherment induces a correspondence between the words in an unknown language and the words in a known language. We focus on the setting where a close correspondence between the alphabets of the two languages exists, but is unknown. Given only two lists of words, the lexicons of both languages, we attempt to induce the correspondence between alphabets and identify the cognates pairs present in the lexicons. The system we propose accomplishes this by defining a simple combinatorial optimization problem that is a function of both the alphabet and cognate matchings, and then induces correspondences by optimizing the objective using a block coordinate descent procedure.

There is a range of past work that has variously investigated cognate detection (Kondrak, 2001; Bouchard-Côté et al., 2007; Bouchard-Côté et al., 2009; Hall and Klein, 2010), character-level decipherment (Knight and Yamada, 1999; Knight et al., 2006; Snyder et al., 2010; Ravi and Knight, 2011), and bilingual lexicon induction (Koehn and Knight, 2002; Haghighi et al., 2008). We consider a common element, which is a model wherein there are character-level correspondences and word-level correspondences, with the word matching parameterized by the character one. This approach subsumes a range of past tasks, though of course past work has specialized in interesting ways.

Past work has emphasized the modeling aspect, where here we use a parametrically simplistic model, but instead emphasize inference.

2 Decipherment as Two-Level Optimization

Our method represents two matchings, one at the alphabet level and one at the lexicon level. A vector of variables x specifies a matching between alphabets. For each character i in the source alphabet and each character j in the target alphabet we define an indicator variable x_{ij} that is on if and only if character i is mapped to character j. Similarly, a vector y represents a matching between lexicons. For word u in the source lexicon and word v in the target lexicon, the indicator variable y_{uv} denotes that u maps to v. Note that the matchings need not be one-to-one.

We define an objective function on the matching variables as follows. Let $\textsc{EditDist}(u, v; x)$ denote the edit distance between source word u and target word v given alphabet matching x. Let the length of word u be l_u and the length of word w be l_w. This edit distance depends on x in the following way. Insertions and deletions always cost a constant ϵ.[1] Substitutions also cost ϵ unless the characters are matched in x, in which case the substitution is

[1] In practice we set $\epsilon = \frac{1}{l_u + l_v}$. $l_u + l_v$ is the maximum number of edit operations between words u and v. This normalization insures that edit distances are between 0 and 1 for all pairs of words.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 313–321,
Edinburgh, Scotland, UK, July 27–31, 2011.

free. Now, the objective that we will minimize can be stated simply: $\sum_u \sum_v y_{uv} \cdot \text{EDITDIST}(u, v; x)$, the sum of the edit distances between the matched words, where the edit distance function is parameterized by the alphabet matching.

Without restrictions on the matchings x and y this objective can always be driven to zero by either mapping all characters to all characters, or matching none of the words. It is thus necessary to restrict the matchings in some way. Let I be the size of the source alphabet and J be the size of the target alphabet. We allow the alphabet matching x to be many-to-many but require that each character participate in no more than two mappings and that the total number of mappings be $\max(I, J)$, a constraint we refer to as *restricted-many-to-many*. The requirements can be encoded with the following linear constraints on x:

$$\forall_i \sum_j x_{ij} \leq 2$$
$$\forall_j \sum_i x_{ij} \leq 2$$
$$\sum_i \sum_j x_{ij} = \max(I, J)$$

The lexicon matching y is required to be τ*-one-to-one*. By this we mean that y is an at-most-one-to-one matching that covers proportion τ of the smaller of the two lexicons. Let U be the size of the source lexicon and V be this size of the target lexicon. This requirement can be encoded with the following linear constraints:

$$\forall_u \sum_v y_{uv} \leq 1$$
$$\forall_v \sum_u y_{uv} \leq 1$$
$$\sum_u \sum_v y_{uv} = \tau \min(U, V)$$

Now we are ready to define the full optimization problem. The first formulation is called the *Implicit Matching Objective* since includes an implicit minimization over edit alignments inside the computation of EDITDIST.

(1) *Implicit Matching Objective*:

$$\min_{x,y} \sum_u \sum_v y_{uv} \cdot \text{EDITDIST}(u, v; x)$$
$$\text{s.t. } x \text{ is restricted-many-to-many}$$
$$y \text{ is } \tau\text{-one-to-one}$$

In order to get a better handle on the shape of the objective and to develop an efficient optimization procedure we decompose each edit distance computation and re-formulate the optimization problem in Section 2.2.

2.1 Example

Figure 1 presents both an example matching problem and a diagram of the variables and objective. Here, the source lexicon consists of the English words `(cat, bat, cart, rat, cab)`, and the source alphabet consists of the characters `(a, b, c, r, t)`. The target alphabet is `(0, 1, 2, 3)`. We have used digits as symbols in the target alphabet to make it clear that we treat the alphabets as disjoint. We have no prior knowledge about any correspondence between alphabets, or between lexicons.

The target lexicon consists of the words `(23, 1233, 120, 323, 023)`. The bipartite graphs show a specific setting of the matching variables. The bold edges correspond to the x_{ij} and y_{uv} that are one. The matchings shown achieve an edit distance of zero between all matched word pairs except for the pair `(cat, 23)`. The best edit alignment for this pair is also diagrammed. Here, '`a`' is aligned to '`2`', '`t`' is aligned to '`3`', and '`c`' is deleted and therefore aligned to the null position '`#`'. Only the initial deletion has a non-zero cost since all other alignments correspond to substitutions between characters that are matched in x.

2.2 Explicit Objective

Computing $\text{EDITDIST}(u, v; x)$ requires running a dynamic program because of the unknown edit alignments; here we define those alignments z explicitly, which makes the $\text{EDITDIST}(u, v; x)$ easy to write explicitly at the cost of more variables. However, by writing the objective in an explicit form that refers to these edit variables, we are able to describe a efficient block coordinate descent procedure that can be used for optimization.

$\text{EDITDIST}(u, v; x)$ is computed by minimizing over the set of monotonic alignments between the characters of the source word u and the characters of the target word v. Let u_n be the character at the nth position of the source word u, and similarly for

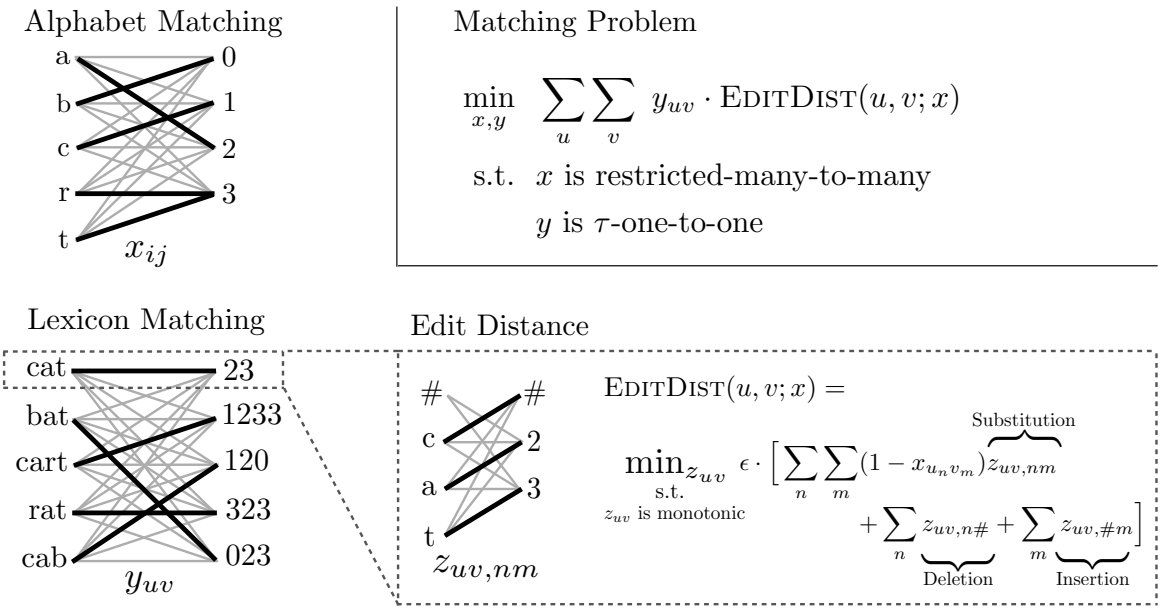


Figure 1: An example problem displaying source and target lexicons and alphabets, along with specific matchings. The variables involved in the optimization problem are diagrammed. x are the alphabet matching indicator variables, y are the lexicon matching indicator variables, and z are the edit alignment indicator variables. The index u refers to a word in the source lexicon, v refers to word in the target lexicon, i refers to a character in the source alphabet, and j refers to a character in the target alphabet. n and m refer to positions in source and target words respectively. The matching objective function is also shown.

v_m. Let z_{uv} be the vector of alignment variables for the edit distance computation between source word u and target word v, where entry $z_{uv,nm}$ indicates whether the character at position n of source word u is aligned to the character at position m of target word v. Additionally, define variables $z_{uv,n\#}$ and $z_{uv,\#m}$ denoting null alignments, which will be used to keep track of insertions and deletions.

$$\text{EDITDIST}(u, v; x) =$$
$$\min_{z_{uv}} \ \epsilon \cdot \Big(\text{SUB}(z_{uv}, x) + \text{DEL}(z_{uv}) + \text{INS}(z_{uv}) \Big)$$
$$\text{s.t. } z_{uv} \text{ is monotonic}$$

We define $\text{SUB}(z_{uv}, x)$ to be the number of substitutions between characters that are not matched in x, $\text{DEL}(z_{uv})$ to be the number of deletions, and $\text{INS}(z_{uv})$ to be the number of insertions.

$$\text{SUB}(z_{uv}, x) = \sum_{n,m} (1 - x_{u_n v_m}) z_{uv,nm}$$
$$\text{DEL}(z_{uv}) = \sum_n z_{uv,n\#}$$
$$\text{INS}(z_{uv}) = \sum_m z_{uv,\#m}$$

Notice that the variable $z_{uv,nm}$ being turned on indicates the substitute operation, while a $z_{uv,n\#}$ or $z_{uv,\#m}$ being turned on indicates an insert or delete operation. These variables are digrammed in Figure 1. The requirement that z_{uv} be a monotonic alignment can be expressed using linear constraints, but in our optimization procedure (described in Section 3) these constraints need not be explicitly represented.

Now we can substitute the explicit edit distance equation into the *implicit matching objective* (1).

Noticing that the mins and sums commute, we arrive at the explicit form of the matching optimization problem.

(2) *Explicit Matching Objective*:

$$\min_{x,y,z} \Big[\sum_{u,v} y_{uv} \cdot \epsilon \cdot \big(\textsc{Sub}(z_{uv}, x) + \textsc{Del}(z_{uv}) + \textsc{Ins}(z_{uv}) \big) \Big]$$

s.t. x is restricted-many-to-many

y is τ-one-to-one

$\forall_{uv}\ z_{uv}$ is monotonic

The implicit and explicit optimizations are the same, apart from the fact that the explicit optimization now explicitly represents the edit alignment variables z. Let the *explicit matching objective* (2) be denoted as $J(x, y, z)$. The relaxation of the explicit problem with 0-1 constraints removed has integer solutions,[2] however the objective $J(x, y, z)$ is non-convex. We thus turn to a block coordinate descent method in the next section in order to find local optima.

3 Optimization Method

We now state a block coordinate descent procedure to find local optima of $J(x, y, z)$ under the constraints on x, y, and z. This procedure alternates between updating y and z to their exact joint optima when x is held fixed, and updating x to its exact optimum when y and z are held fixed.

The psuedocode for the procedure is given in Algorithm 1. Note that the function EDITDIST returns both the min edit distance e_{uv} and the argmin edit alignments z_{uv}. Also note that c_{ij} is as defined in Section 3.2.

3.1 Lexicon Matching Update

Let x, the alphabet matching variable, be fixed. We consider the problem of optimizing $J(x, y, z)$ over the lexicon matching variable y and and the edit alignments z under the constraint that y is τ-one-to-one and each z_{uv} is monotonic.

[2]This can be shown by observing that optimizing x when y and z are held fixed yields integer solutions (shown in Section 3.2), and similarly for the optimization of y and z when x is fixed (shown in Section 3.1). Thus, every local optimum with respect to these block coordinate updates has integer solutions. The global optimum must be one of these local optima.

Algorithm 1 Block Coordinate Descent

Randomly initialize alphabet matching x.
repeat
  for all u, v **do**
    $(e_{uv}, z_{uv}) \leftarrow \textsc{EditDist}(u, v; x)$
  end for
  [Hungarian]
  $y \leftarrow \text{argmin}_{y\ \tau\text{-one-to-one}} \Big[\sum_{u,v} y_{uv} e_{uv} \Big]$
  [Solve LP]
  $x \leftarrow \text{argmax}_{x\ \text{restr.-many-to-many}} \Big[\sum_{i,j} x_{ij} c_{ij} \Big]$
until convergence

Notice that y simply picks out which edit distance problems affect the objective. The z_{uv} in each of these edit distance problems can be optimized independently. z_{uv} that do not have y_{uv} active have no effect on the objective, and z_{uv} with y_{uv} active can be optimized using the standard edit distance dynamic program. Thus, in a first step we compute the $U \cdot V$ edit distances e_{uv} and best monotonic alignment variables z_{uv} between all pairs of source and target words using $U \cdot V$ calls to the standard edit distance dynamic program. Altogether, this takes time $O\big((\sum_u l_u) \cdot (\sum_v l_v)\big)$.

Now, in a second step we compute the least weighted τ-one-to-one matching y under the weights e_{uv}. This can be accomplished in time $O(\max(U, V)^3)$ using the Hungarian algorithm (Kuhn, 1955). These two steps produce y and z that exactly achieve the optimum value of $J(x, y, z)$ for the given value of x.

3.2 Alphabet Matching Update

Let y and z, the lexicon matching variables and the edit alignments, be fixed. Now, we find the optimal alphabet matching variables x subject to the constraint that x is restricted-many-to-many.

It makes sense that to optimize $J(x, y, z)$ with respect to x we should prioritize mappings x_{ij} that would mitigate the largest substitution costs in the active edit distance problems. Indeed, with a little algebra it can be shown that solving a maximum weighted matching problem with weights c_{ij} that count potential substitution costs gives the correct update for x. In particular, c_{ij} is the total cost of substitution edits in the active edit alignment prob-

lems that would result if source character i were not mapped to target character j in the alphabet matching x. This can be written as:

$$c_{ij} = \sum_{u,v} \sum_{n,m \text{ s.t. } u_n=i, v_m=j} \epsilon \cdot y_{uv} \cdot z_{uv,nm}$$

If x were constrained to be one-to-one, we could again apply the Hungarian algorithm, this time to find a maximum weighted matching under the weights c_{ij}. Since we have instead allowed restricted-many-to-many alphabet matchings we turn to linear programming for optimizing x. We can state the update problem as the following linear program (LP), which is guaranteed to have integer solutions:

$$\begin{aligned} \min_{x} \quad & \sum_{ij} x_{ij} c_{ij} \\ \text{s.t.} \quad & \forall_i \sum_j x_{ij} \leq 2, \quad \forall_j \sum_i x_{ij} \leq 2 \\ & \sum_i \sum_j x_{ij} = \max(I, J) \end{aligned}$$

In experiments we used the GNU Linear Programming Toolkit (GLPK) to solve the LP and update the alphabet matching x. This update yields matching variables x that achieve the optimum value of $J(x, y, z)$ for fixed y and z.

3.3 Random Restarts

In practice we found that the block coordinate descent procedure can get stuck at poor local optima. To find better optima, we run the coordinate descent procedure multiple times, initialized each time with a random alphabet matching. We choose the local optimum with the best objective value across all initializations. This approach yielded substantial improvements in achieved objective value.

4 Experiments

We compare our system to three different state-of-the-art systems on three different data sets. We set up experiments that allow for as direct a comparison as possible. In some cases it must be pointed out that the past system's goals are different from our own, and we will be comparing in a different way than the respective work was intended. The three systems make use of additional, or slightly different, sources of information.

4.1 Phonetic Cognate Lexicons

The first data set we evaluate on consists of 583 triples of phonetic transcriptions of cognates in Spanish, Portuguese, and Italian. The data set was introduced by Bouchard-Côté et al. (2007). For a given pair of languages the task is to determine the mapping between lexicons that correctly maps each source word to its cognate in the target lexicon. We refer to this task and data set as ROMANCE.

Hall and Klein (2010) presented a state-of-the-art system for the task of cognate identification and evaluated on this data set. Their model explicitly represents parameters for phonetic change between languages and their parents in a phylogenetic tree. They estimate parameters and infer the pairs of cognates present in all three languages jointly, while we consider each pair of languages in turn.

Their model has similarities with our own in that it learns correspondences between the alphabets of pairs of languages. However, their correspondences are probabilistic and implicit while ours are hard and explicit. Their model also differs from our own in a key way. Notice that the phonetic alphabets for the three languages are actually the same. Since phonetic change occurs gradually across languages a helpful prior on the correspondence is to favor the identity. Their model makes use of such a prior. Our model, on the other hand, is unaware of any prior correspondence between alphabets and does not make use of this additional information about phonetic change.

Hall and Klein (2010) also evaluate their model on lexicons that do not have a perfect cognate mapping. This scenario, where not every word in one language has a cognate in another, is more realistic. They produced a data set with this property by pruning words from the ROMANCE data set until only about 75% of the words in each source lexicon have cognates in each target lexicon. We refer to this task and data set as PARTIALROMANCE.

4.2 Lexicons Extracted from Corpora

Next, we evaluate our model on a noisier data set. Here the lexicons in source and target languages are extracted from corpora by taking the top 2,000 words in each corpus. In particular, we used the English and Spanish sides of the Europarl parallel cor-

pus (Koehn, 2005). To make this set up more realistic (though fairly comparable), we insured that the corpora were non-parallel by using the first 50K sentences on the English side and the second 50K sentences on the Spanish side. To generate a gold cognate matching we used the intersected HMM alignment model of Liang et al. (2008) to align the full parallel corpus. From this alignment we extracted a translation lexicon by adding an entry for each word pair with the property that the English word was aligned to the Spanish in over 10% of the alignments involving the English word. To reduce this translation lexicon down to a cognate matching we went through the translation lexicon by hand and removed any pair of words that we judged to not be cognates. The resulting gold matching contains cognate mappings in the English lexicon for 1,026 of the words in the Spanish lexicon. This means that only about 50% of the words in English lexicon have cognates in the Spanish lexicon. We evaluate on this data set by computing precision and recall for the number of English words that are mapped to a correct cognate. We refer to this task and data set as EUROPARL.

On this data set, we compare against the state-of-the-art orthographic system presented in Haghighi et al. (2008). Haghighi et al. (2008) presents several systems that are designed to extract translation lexicons for non-parallel corpora by learning a correspondence between their monolingual lexicons. Since our system specializes in matching cognates and does not take into account additional information from corpus statistics, we compare against the version of their system that only takes into account orthographic features and is thus is best suited for cognate detection. Their system requires a small seed of correct cognate pairs. From this seed the system learns a projection using canonical correlation analysis (CCA) into a canonical feature space that allows feature vectors from source words and target words to be compared. Once in this canonical space, similarity metrics can be computed and words can be matched using a bipartite matching algorithm. The process is iterative, adding cognate pairs to the seed lexicon gradually and each time re-computing a refined projection. Our system makes no use of a seed lexicon whatsoever.

Both our system and the system of Haghighi et al. (2008) must solve bipartite matching problems between the two lexicons. For this data set, the lexicons are large enough that finding the exact solution can be slow. Thus, in all experiments on this data set, we instead use a greedy competitive linking algorithm that runs in time $O(U^2V^2\log(UV))$.

Again, for this dataset it is reasonable to expect that many characters will map to themselves in the best alphabet matching. The alphabets are not identical, but are far from disjoint. Neither our system, nor that of Haghighi et al. (2008) make use of this expectation. As far as both systems are concerned, the alphabets are disjoint.

4.3 Decipherment

Finally, we evaluate our model on a data set where a main goal is to decipher an unknown correspondence between alphabets. We attempt to learn a mapping from the alphabet of the ancient Semitic language Ugaritic to the alphabet of Hebrew, and at the same time learn a matching between Hebrew words in a Hebrew lexicon and their cognates in a Ugaritic lexicon. This task is related to the task attempted by Snyder et al. (2010). The data set consists of a Ugaritic lexicon of 2,214 words, each of which has a Hebrew cognate, the lexicon of their 2,214 Hebrew cognates, and a gold cognate dictionary for evaluation. We refer to this task and data set as UGARITIC.

The non-parameteric Bayesian system of Snyder et al. (2010) assumes that the morphology of Hebrew is known, making use of an inventory of suffixes, prefixes, and stems derived from the words in the Hebrew bible. It attempts to learn a correspondence between the morphology of Ugaritic and that of Hebrew while reconstructing cognates for Ugaritic words. This is a slightly different goal than that of our system, which learns a correspondence between lexicons. Snyder et al. (2010) run their system on a set 7,386 Ugaritic words, the same set that we extracted our 2,214 Ugaritic words with Hebrew cognates from. We evaluate the accuracy of the lexicon matching produced by our system on these 2,214 Ugaritic words, and so do they, measuring the number of correctly reconstructed cognates.

By restricting the source and target lexicons to sets of cognates we have made the task easier. This was necessary, however, because the Ugaritic and Hebrew corpora used by Snyder et al. (2010) are not

Model	τ	Accuracy
Hall and Klein (2010)	–	**90.3**
MATCHER	1.0	90.1

Table 1: Results on ROMANCE data set. Our system is labeled MATCHER. We compare against the phylogenetic cognate detection system of Hall and Klein (2010). We show the pairwise cognate accuracy across all pairs of languages from the following set: Spanish, Portuguese, and Italian.

comparable: only a small proportion of the words in the Ugaritic lexicon have cognates in the lexicon composed of the most frequent Hebrew words.

Here, the alphabets really are disjoint. The symbols in both languages look nothing alike. There is no obvious prior expectation about how the alphabets will be matched. We evaluate against a well-established correspondence between the alphabets of Ugaritic and Hebrew. The Ugaritic alphabet contains 30 characters, the Hebrew alphabet contains 22 characters, and the gold matching contains 33 entries. We evaluate the learned alphabet matching by counting the number of recovered entries from the gold matching.

Due to the size of the source and target lexicons, we again use the greedy competitive linking algorithm in place of the exact Hungarian algorithm in experiments on this data set.

5 Results

We present results on all four datasets ROMANCE, PARTIALROMANCE, EUROPARL, and UGARITIC. On the ROMANCE and PARTIALROMANCE data sets we compare against the numbers published by Hall and Klein (2010). We ran an implementation of the orthographic system presented by Haghighi et al. (2008) on our EUROPARL data set. We compare against the numbers published by Snyder et al. (2010) on the UGARITIC data set. We refer to our system as MATCHER in result tables and discussion.

5.1 ROMANCE

The results of running our system, MATCHER, on the ROMANCE data set are shown in Table 1. We recover 88.9% of the correct cognate mappings on the pair Spanish and Italian, 85.7% on Italian and Portuguese, and 95.6% on Spanish and Portuguese.

Model	τ	Precision	Recall	F1
Hall and Klein (2010)	–	66.9	**82.0**	73.6
MATCHER	0.25	**99.7**	34.0	50.7
	0.50	93.8	60.2	73.3
	0.75	81.1	78.0	**79.5**

Table 2: Results on PARTIALROMANCE data set. Our system is labeled MATCHER. We compare against the phylogenetic cognate detection system of Hall and Klein (2010). We show the pairwise cognate precision, recall, and F1 across all pairs of languages from the following set: Spanish, Portuguese, and Italian. Note that approximately 75% of the source words in each of the source lexicons have cognates in each of the target lexicons.

Our average accuracy across all pairs of languages is 90.1%. The phylogenetic system of Hall and Klein (2010) achieves an average accuracy of 90.3% across all pairs of languages. Our system achieves accuracy comparable to that of the phylogenetic system, despite the fact that the phylogenetic system is substantially more complex and makes use of an informed prior on alphabet correspondences.

The alphabet matching learned by our system is interesting to analyze. For the pairing of Spanish and Portuguese it recovers phonetic correspondences that are well known. Our system learns the correct cognate pairing of Spanish /bino/ to Portuguese /vinu/. This pair exemplifies two common phonetic correspondences for Spanish and Portuguese: the Spanish /o/ often transforms to a /u/ in Portuguese, and Spanish /b/ often transforms to /v/ in Portuguese. Our system, which allows many-to-many alphabet correspondences, correctly identifies the mappings /o/ $\rightarrow$ /u/ and /b/ $\rightarrow$ /v/ as well as the identity mappings /o/ $\rightarrow$ /o/ and /b/ $\rightarrow$ /b/ which are also common.

5.2 PARTIALROMANCE

In Table 2 we present the results of running our system on the PARTIALROMANCE data set. In this data set, only approximately 75% of the source words in each of the source lexicons have cognates in each of the target lexicons. The parameter τ trades off precision and recall. We show results for three different settings of τ: 0.25, 0.5, and 0.75.

Our system achieves an average precision across language pairs of 99.7% at an average recall of 34.0%. For the pairs Italian – Portuguese, and Span-

Model	Seed	τ	Precision	Recall	F1
Haghighi et al. (2008)	20	0.1	72.0	14.0	23.5
	20	0.25	63.6	31.0	41.7
	20	0.5	44.8	43.7	44.2
	50	0.1	90.5	17.6	29.5
	50	0.25	75.4	36.7	49.4
	50	0.5	56.4	55.0	55.7
MATCHER	0	0.1	**93.5**	18.2	30.5
	0	0.25	83.2	40.5	54.5
	0	0.5	56.5	**55.1**	**55.8**

Table 3: Results on EUROPARL data set. Our system is labeled MATCHER. We compare against the bilingual lexicon induction system of Haghighi et al. (2008). We show the cognate precision, recall, and F1 for the pair of languages English and Spanish using lexicons extracted from corpora. Note that approximately 50% of the words in the English lexicon have cognates in the Spanish lexicon.

ish – Portuguese, our system achieves prefect precision at recalls of 32.2% and 38.1% respectively. The best average F1 achieved by our system is 79.5%, which surpasses the average F1 of 73.6 achieved by the phylogenetic system of Hall and Klein (2010).

The phylogenetic system observes the phylogenetic tree of ancestry for the three languages and explicitly models cognate evolution and survival in a 'survival' tree. One might expect the phylogenetic system to achieve better results on this data set where part of the task is identifying which words do not have cognates. It is surprising that our model does so well given its simplicity.

5.3 EUROPARL

Table 3 presents results for our system on the EUROPARL data set across three different settings of τ: 0.1, 0.25, and 0.5. We compare against the orthographic system presented by Haghighi et al. (2008), across the same three settings of τ, and with two different sizes of seed lexicon: 20 and 50. In this data set, only approximately 50% of the source words have cognates in the target lexicon.

Our system achieves a precision of 93.5% at a recall of 18.2%, and a best F1 of 55.0%. Using a seed matching of 50 word pairs, the orthographic system of Haghighi et al. (2008) achieves a best F1 of 55.7%. Using a seed matching of 20 word pairs, it achieves a best F1 of 44.2%. Our system outperforms the orthographic system even though the orthographic system makes use of important additional information: a seed matching of correct cognate pairs. The results show that as the size of this seed is decreased, the performance of the orthographic system degrades.

Model	τ	Lexicon Acc.	Alphabet Acc.
Snyder et al. (2010)	–	60.4*	**29/33***
MATCHER	1.0	**90.4**	28/33

Table 4: Results on UGARITIC data set. Our system is labeled MATCHER. We compare against the decipherment system of Snyder et al. (2010). *Note that results for this system are on a somewhat different task. In particular, the MATCHER system assumes the inventories of cognates in both Hebrew and Ugaritic are known, while the system of Snyder et al. (2010) *reconstructs* cognates assuming only that the morphology of Hebrew is known, which is a harder task. We show cognate pair identification accuracy and alphabet matching accuracy for Ugaritic and Hebrew.

5.4 UGARITIC

In Table 4 we present results on the UGARITIC data set. We evaluate both accuracy of the lexicon matching learned by our system, and the accuracy of the alphabet matching. Our system achieves a lexicon accuracy of 90.4% while correctly identifying 28 out the 33 gold character mappings.

We also present the results for the decipherment model of Snyder et al. (2010) in Table 4. Note that while the evaluation data sets for our two models are the same, the tasks are very different. In particular, our system assumes the inventories of cognates in both Hebrew and Ugaritic are known, while the system of Snyder et al. (2010) reconstructs cognates assuming only that the morphology of Hebrew is known, which is a harder task. Even so, the results show that our system is effective at decipherment when semantically similar lexicons are available.

6 Conclusion

We have presented a simple combinatorial model that simultaneously incorporates both a matching between alphabets and a matching between lexicons. Our system is effective at both the tasks of cognate identification and alphabet decipherment, requiring only lists of words in both languages as input.

References

A. Bouchard-Côté, P. Liang, T.L. Griffiths, and D. Klein. 2007. A probabilistic approach to diachronic phonology. In *Proc. of EMNLP*.

A. Bouchard-Côté, T.L. Griffiths, and D. Klein. 2009. Improved reconstruction of protolanguage word forms. In *Proc. of NAACL*.

A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein. 2008. Learning bilingual lexicons from monolingual corpora. *Proceedings of ACL*.

D. Hall and D. Klein. 2010. Finding cognate groups using phylogenies. In *Proc. of ACL*.

K. Knight and K. Yamada. 1999. A computational approach to deciphering unknown scripts. In *Proc. of ACL Workshop on Unsupervised Learning in Natural Language Processing*.

K. Knight, A. Nair, N. Rathod, and K. Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proc. of COLING/ACL*.

P. Koehn and K. Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proc. of ACL workshop on Unsupervised lexical acquisition*.

P. Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proc. of Machine Translation Summit*.

G. Kondrak. 2001. Identifying Cognates by Phonetic and Semantic Similarity. In *NAACL*.

H.W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*.

P. Liang, D. Klein, and M.I. Jordan. 2008. Agreement-based learning. *Proc. of NIPS*.

S. Ravi and K. Knight. 2011. Bayesian inference for Zodiac and other homophonic ciphers. In *Proc. of ACL*.

B. Snyder, R. Barzilay, and K. Knight. 2010. A statistical model for lost language decipherment. In *Proc. of ACL*.

Universal Morphological Analysis using Structured Nearest Neighbor Prediction

Young-Bum Kim
University of Wisconsin-Madison
ybkim@cs.wisc.edu

João V. Graça
L^2F INESC-ID
Lisboa, Portugal
joao.graca@l2f.inesc-id.pt

Benjamin Snyder
University of Wisconsin-Madison
bsnyder@cs.wisc.edu

Abstract

In this paper, we consider the problem of unsupervised morphological analysis from a new angle. Past work has endeavored to design unsupervised learning methods which explicitly or implicitly encode inductive biases appropriate to the task at hand. We propose instead to treat morphological analysis as a structured prediction problem, where languages with labeled data serve as training examples for unlabeled languages, without the assumption of parallel data. We define a universal morphological feature space in which every language and its morphological analysis reside. We develop a novel structured nearest neighbor prediction method which seeks to find the morphological analysis for each unlabeled language which lies as close as possible in the feature space to a training language. We apply our model to eight inflecting languages, and induce nominal morphology with substantially higher accuracy than a traditional, MDL-based approach. Our analysis indicates that accuracy continues to improve substantially as the number of training languages increases.

1 Introduction

Over the past several decades, researchers in the natural language processing community have focused most of their efforts on developing text processing tools and techniques for English (Bender, 2009), a morphologically simple language. Recently, increasing attention has been paid to the wide variety of other languages of the world. Most of these languages still pose severe difficulties, due to (i) their lack of annotated textual data, and (ii) the fact that they exhibit linguistic structure not found in English, and are thus not immediately susceptible to many traditional NLP techniques.

Consider the example of nominal part-of-speech analysis. The Penn Treebank defines only four English noun tags (Marcus et al., 1994), and as a result, it is easy to treat the words bearing these tags as completely distinct word classes, with no internal morphological structure. In contrast, a comparable tagset for Hungarian includes 154 distinct noun tags (Erjavec, 2004), reflecting Hungarian's rich inflectional morphology. When dealing with such languages, treating words as atoms leads to severe data sparsity problems.

Because annotated resources do not exist for most morphologically rich languages, prior research has focused on unsupervised methods, with a focus on developing appropriate inductive biases. However, inductive biases and declarative knowledge are notoriously difficult to encode in well-founded models. Even putting aside this practical matter, a universally correct inductive bias, if there is one, is unlikely to be be discovered by *a priori* reasoning alone.

In this paper, we argue that languages for which we *have* gold-standard morphological analyses can be used as effective guides for languages *lacking* such resources. In other words, instead of treating each language's morphological analysis as a *de novo* induction problem to be solved with a purely hand-coded bias, we instead learn from our labeled languages what linguistically plausible morphological analyses looks like, and guide our analysis in this direction.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 322–332,
Edinburgh, Scotland, UK, July 27–31, 2011.

More formally, we recast morphological induction as a new kind of supervised structured prediction problem, where each annotated language serves as a single training example. Each language's noun lexicon serves as a single input x, and the analysis of the nouns into stems and suffixes serves as a complex structured label y.

Our first step is to define a universal morphological feature space, into which each language and its morphological analysis can be mapped. We opt for a simple and intuitive mapping, which measures the sizes of the stem and suffix lexicons, the entropy of these lexicons, and the fraction of word forms which appear without any inflection.

Because languages tend to cluster into well defined morphological groups, we cast our learning and prediction problem in the nearest neighbor framework (Cover and Hart, 1967). In contrast to its typical use in classification problems, where one can simply pick the label of the nearest training example, we are here faced with a structured prediction problem, where locations in feature space depend jointly on the input-label pair (x, y). Finding a nearest neighbor thus consists of *searching* over the space of morphological analyses, until a point in feature space is reached which lies closest to one of the labeled languages. See Figure 1 for an illustration.

To provide a measure of empirical validation, we applied our approach to eight languages with inflectional nominal morphology, ranging in complexity from very simple (English) to very complex (Hungarian). In all but one case, our approach yields substantial improvements over a comparable monolingual baseline (Goldsmith, 2005), which uses the minimum description length principle (MDL) as its inductive bias. On average, our method increases accuracy by 11.8 percentage points, corresponding to a 42% decrease in error relative to a supervised upper bound. Further analysis indicates that accuracy improves as the number of training languages increases.

2 Related Work

In this section, we briefly review prior work on unsupervised morphological induction, as well as multilingual analysis in NLP.

Unsupervised Morphological Induction: Unsupervised morphology remains an active area of research (Schone and Jurafsky, 2001; Goldsmith, 2005; Adler and Elhadad, 2006; Creutz and Lagus, 2005; Dasgupta and Ng, 2007; Creutz and Lagus, 2007; Poon et al., 2009). Many existing algorithms derive morpheme lexicons by identifying recurring patterns in words. The goal is to optimize the compactness of the data representation by finding a small lexicon of highly frequent strings, resulting in a *minimum description length* (MDL) lexicon and corpus (Goldsmith, 2001; Goldsmith, 2005). Later work cast this idea in a probabilistic framework in which the the MDL solution is equivalent to a MAP estimate in a suitable Bayesian model (Creutz and Lagus, 2005). In all these approaches, a locally optimal segmentation is identified using a task-specific greedy search.

Multilingual Analysis: An influential line of prior multilingual work starts with the observation that rich linguistic resources exist for some languages but not others. The idea then is to *project* linguistic information from one language onto others via parallel data. Yarowsky and his collaborators first developed this idea and applied it to the problems of part-of-speech tagging, noun-phrase bracketing, and morphology induction (Yarowsky and Wicentowski, 2000; Yarowsky et al., 2000; Yarowsky and Ngai, 2001), and other researchers have applied the idea to syntactic and semantic analysis (Hwa et al., 2005; Padó and Lapata, 2006) In these cases, the existence of a bilingual parallel text along with highly accurate predictions for one of the languages was assumed.

Another line of work assumes the existence of bilingual parallel texts without the use of any supervision (Dagan et al., 1991; Resnik and Yarowsky, 1997). This idea has been developed and applied to a wide variety tasks, including morphological analysis (Snyder and Barzilay, 2008b; Snyder and Barzilay, 2008a), part-of-speech induction (Snyder et al., 2008; Snyder et al., 2009b; Naseem et al., 2009), and grammar induction (Snyder et al., 2009a; Blunsom et al., 2009; Burkett et al., 2010). An even more recent line of work does away with the assumption of parallel texts and performs joint unsupervised induction for various languages through the use of coupled priors in the context of grammar in-

duction (Cohen and Smith, 2009; Berg-Kirkpatrick and Klein, 2010).

In contrast to these previous approaches, the method proposed in this paper does *not* assume the existence of any parallel text, but *does* assume that labeled data exists for a wide variety of languages, to be used as training examples for our test language.

3 Structured Nearest Neighbor

We reformulate morphological induction as a *supervised* learning task, where each annotated language serves as a single training example for our language-independent model. Each such example consists of an input-label pair (x, y), both of which contain complex internal structure: The input $x \in \mathcal{X}$ consists of a vocabulary list of all words observed in a particular monolingual corpus, and the label $y \in \mathcal{Y}$ consists of the correct morphological analysis of all the vocabulary items in x.[1] Because our goal is to generalize across languages, we define a feature function which maps each (x, y) pair to a universal feature space: $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$.

For each unlabeled input language x, our goal is to predict a complete morphological analysis $y \in \mathcal{Y}$ which maximizes a scoring function on the feature space, $score : \mathbb{R}^d \rightarrow \mathbb{R}$. This scoring function is trained using the n labeled-language examples: $(x, y)_1, \ldots, (x, y)_n$, and the resulting prediction rule for unlabeled input x is given by:

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, score\big(\mathbf{f}(x, y)\big)$$

Languages can be typologically categorized by the type and richness of their morphology. On the assumption that for each test language, at least one typologically similar language will be present in the training set, we employ a *nearest neighbor* scoring function. In the standard nearest neighbor classification setting, one simply predicts the label of the closest training example in the input space.[2] In our structured prediction setting, the mapping to the universal feature space depends crucially on the structure of the proposed label y, not simply the input x. We thus generalize nearest-neighbor prediction to the structured scenario and propose the following prediction rule:

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \min_{\ell} \parallel \mathbf{f}(x, y) - \mathbf{f}(x_\ell, y_\ell) \parallel, \quad (1)$$

where the index ℓ ranges over the training languages. In words, we predict the morphological analysis y for our test language which places it as close as possible in the universal feature space to one of the training languages ℓ.

Morphological Analysis: In this paper we focus on nominal inflectional suffix morphology. Consider the word *utiskom* in Serbian, meaning *impression* with the instrumental case marking. A correct analysis of this word would divide it into a stem (*utisak* = *impression*), a suffix (*-om* = instrumental case), and a phonological deletion rule on the stem's penultimate vowel ($..ak\# \rightarrow ..k\#$).

More generally, as we define it, a morphological analysis of a word type w consists of (i) a stem t, (ii), a suffix f, and (iii) a deletion rule d. Either or both of the suffix and deletion rule can be $NULL$. We allow three types of deletion rules on stems: deletion of final vowels ($..V\# \rightarrow ..\#$), deletion of penultimate vowels ($..VC\# \rightarrow ..C\#$), and removals and additions of final accent marks (e.g. $..\tilde{a}\# \rightarrow ..a\#$). We require that stems be at least three characters long and that suffixes be no more than four. And, of course, we require that after (1) applying deletion rule d to stem t, and (2) adding suffix f to the result, we obtain word w.

Universal Feature Space: We employ a fairly simple and minimal set of features, all of which could plausibly generalize across a wide range of languages. Consider the set of stems T, suffixes F, and deletion rules D, induced by the morphological analyses y of the words x. Our first three features simply count the sizes of these three sets.

These counting features consider only the raw number of unique morphemes (and phonological rules) being used, but not their individual frequency or distribution. Our next set of features considers the empirical *entropy* of these occurrences as distributed across the lexicon of words x by analysis y.

[1] Technically, the label space of each input, $\mathcal{Y}$, should be thought of as a function of the input x. We suppress this dependence for notational clarity.

[2] More generally the majority label of the k-nearest neighbors.

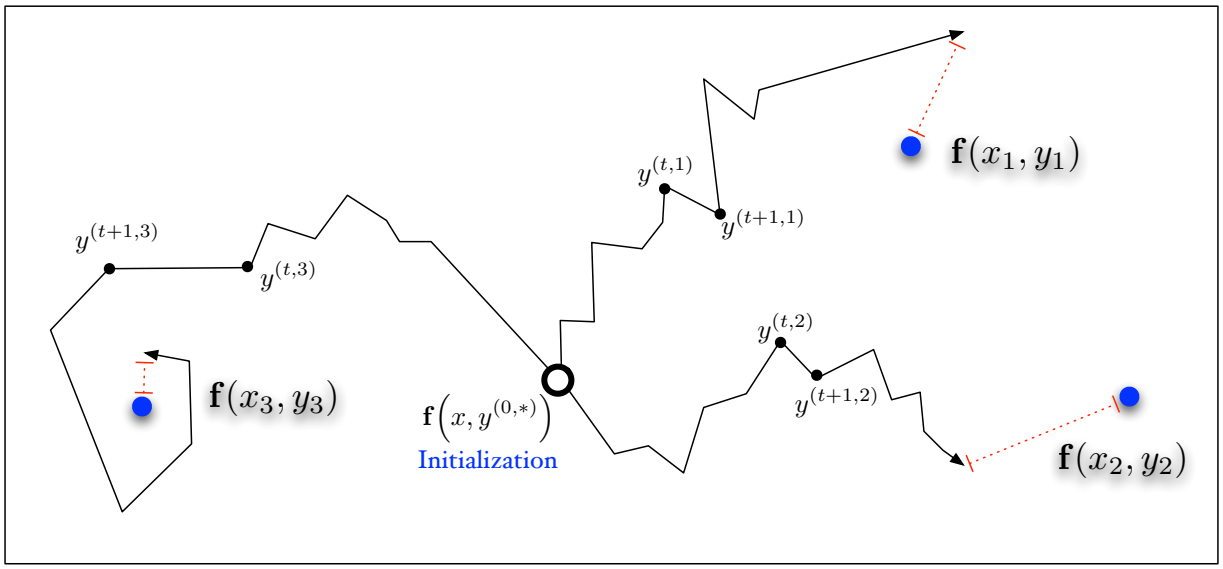


Figure 1: **Structured Nearest Neighbor Search:** The inference procedure for unlabeled test language x, when trained with three labeled languages, $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Our search procedure iteratively attempts to find labels for x which are as close as possible in feature space to each of the training languages. After convergence, the label which is closest in distance to a training language is predicted, in this case being the label near training language (x_3, y_3).

For example, if the (x, y) pair consists of the analyzed words {*kiss, kiss-es, hug*}, then the empirical distributions over stems, suffixes, and deletion rules would be:

- $P(t = kiss) = 2/3$
- $P(t = hug) = 1/3$
- $P(f = NULL) = 2/3$
- $P(f = -es) = 1/3$
- $P(d = NULL) = 1$

The three entropy features are defined as the shannon entropies of these stem, suffix, and deletion rule probabilities: $H(t), H(f), H(d)$.[3]

Finally, we consider two simple *percentage* features: the percentage of words in x which according to y are left unsegmented (i.e. have the null suffix, 2/3 in the example above), and the percentage of segmented words which employ a deletion rule (0 in the example above). Thus, in total, our model employs 8 universal morphological features. All features are scaled to the unit interval and are assumed to have equal weight.

[3]Note that here and throughout the paper, we operate over word types, ignoring their corpus frequencies.

3.1 Search Algorithm

The main algorithmic challenge for our model lies in efficiently computing the best morphological analysis y for each language-specific word set x, according to Equation 1. Exhaustive search through the set of all possible morphological analyses is impossible, as the number of such analyses grows exponentially in the size of the vocabulary. Instead, we develop a greedy search algorithm in the following fashion (the search procedure is visually depicted in Figure 1).

At each time-step t, we maintain a set of frontier analyses $\left\{y^{(t,\ell)}\right\}_\ell$, where ℓ ranges over the training languages. The goal is to iteratively modify each of these frontier analyses $y^{(t,\ell)} \rightarrow y^{(t+1,\ell)}$ so that the location of the training language in universal feature space — $\mathbf{f}\left(x, y^{(t+1,\ell)}\right)$ — is as close as possible to the location of the training language ℓ: $\mathbf{f}\left(x_\ell, y_\ell\right)$.

After iterating this procedure to convergence, we are left with a set of analyses $\left\{y^{(\ell)}\right\}_\ell$, each of which approximates the analyses which yield minimal distances to a particular training language:

$$y^{(\ell)} \approx \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \parallel \mathbf{f}(x, y) - \mathbf{f}(x_\ell, y_\ell) \parallel .$$

We finally select from amongst these analyses and

make our prediction:

$$\ell^* = \operatorname*{argmin}_{\ell} \parallel \mathbf{f}(x, y^{(\ell)}) - \mathbf{f}(x_\ell, y_\ell) \parallel$$
$$y^* = y^{(\ell^*)}$$

The main outline of our search algorithm is based on the MDL-based greedy search heuristic developed and studied by (Goldsmith, 2005). At a high level, this search procedure alternates between individual analyses of words (keeping the set of stems and suffixes fixed), aggregate discoveries of new stems (keeping the suffixes fixed), and aggregate discoveries of new suffixes (keeping stems fixed). As input, we consider the test words x in our new language, and we run the search in parallel for each training language (x_ℓ, y_ℓ). For each such test-train language pair, the search consists of the following stages:

Stage 0: Initialization

We initially analyze each word $w \in x$ according to peaks in *successor frequency*.[4] If w's n-character prefix $w_{:n}$ has successor frequency > 1 and the surrounding prefixes, $w_{:n-1}$ and $w_{:n+1}$ both have successor frequency $= 1$, then we analyze w as a stem-suffix pair: $(w_{:n}, w_{n+1:})$.[5] Otherwise, we initialize w as an unsuffixed stem. As this procedure tends to produce an overly large set of suffixes F, we further prune F down to the number of suffixes found in the training language, retaining those which appear with the largest number of stems. This initialization stage is carried out once, and afterwards the following three stages are repeated until convergence.

Stage 1: Reanalyze each word

In this stage, we reanalyze each word (in random order). We use the set of stems T and suffixes F obtained from the previous stage, and don't permit the addition of any new items to these lists. Instead, we focus on obtaining better analyses of each word, while also building up a set of phonological deletion rules D. For each word $w \in x$, we consider all possible segmentations of w into a stem-suffix pair (t, f), for which $f \in F$, and where either $t \in T$ or some $t' \in T$ such that t is obtained from t' using a deletion rule d (e.g. by deleting a final or penultimate vowel). For each such possible analysis y', we compute the resulting location in feature space $\mathbf{f}(x, y')$, and select the analysis that brings us closest to our target training language: $y = \operatorname{argmin}_{y'} \parallel \mathbf{f}(x, y') - \mathbf{f}(x_\ell, y_\ell) \parallel$.

Stage 2: Find New Stems

In this stage, we keep our set of suffixes F and deletion rules D from the previous stage fixed, and attempt to find new stems to add to T through an aggregate analysis of unsegmented words. For every string s, we consider the set of words which are currently unsegmented, and can be analyzed as a stem-suffix pair (s, f) for some existing suffix $f \in F$, and some deletion rule $d \in D$. We then consider the joint segmentation of these words into a new stem s, and their respective suffixes. As before, we choose the segmentation if it brings us closer in feature space to our target training language.

Stage 3: Find New Suffixes

This stage is exactly analogous to the previous stage, except we now fix the set of stems T and seek to find new suffixes.

3.2 A Monolingual Supervised Model

In order to provide a plausible upper bound on performance, we also formulate a supervised monolingual morphological model, using the structured perceptron framework (Collins, 2002). Here we assume that we are given some training sequence of inputs and morphological analyses (all within one language): $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. We define each input x_i to be a noun w, along with a morphological tag z, which specifies the gender, case, and number of the noun. The goal is to predict the correct segmentation of w into stem, suffix, and phonological deletion rule: $y_i = (t, f, d)$.[6]

To do so, we define a feature function over input-label pairs, (x, y), with the following binary feature templates: (1) According to label y_i, the stem is t

[4] The *successor frequency* of a string prefix s is defined as the number of unique characters that occur immediately after s in the vocabulary.

[5] With the restriction that at this stage we only allow suffixes up to length 5, and stems of at least length 3.

[6] While the assumption of the correct morphological tag as input is somewhat unrealistic, this model still gives us a strong upper bound on how well we can expect our unsupervised model to perform.

	Type Counts				Entropy			Percentage	
	# words	# stems	# suffs	# dels	stem entropy	suff entropy	del entropy	unseg	deleted
BG	4833	3112	21	8	11.4	2.7	0.9	.45	.29
CS	5836	3366	28	12	11.5	3.2	1.6	.38	.53
EN	4178	3453	3	1	11.7	1.0	0.1	.73	.06
ET	6371	3742	141	5	11.5	5.0	0.2	.31	.04
HU	8051	3746	231	7	11.3	5.8	0.5	.23	.11
RO	5578	3297	23	8	11.5	2.9	1.4	.48	.51
SL	6111	3172	32	6	11.3	3.2	1.5	.33	.56
SR	5849	3178	28	5	11.4	2.9	1.4	.33	.53

Table 1: Corpus statistics for the eight languages. The first four columns give the number of unique word, stem, suffix, and phonological deletion rule types. The next three columns give, respectively, the entropies of the distributions of stems, suffixes (including $NULL$), and deletion rules (including $NULL$) over word types. The final two columns give, respectively, the percentage of word types occurring with the $NULL$ suffix, and the number of non-$NULL$ suffix words which use a phonological deletion rule. Note that the final eight columns define the universal feature space used by our model. BG = Bulgarian, CS = Czech, EN = English, ET = Estonian, HU = Hungarian, RO = Romanian, SL = Slovene, SR = Serbian

(one feature for each possible stem). (2) According to label y_i, the suffix and deletion rule are (f, d) (one feature for every possible pair of deletion rules and suffixes). (3) According to label y_i and morphological tag z, the suffix, deletion rule, and gender are respectively (f, d, G). (4) According to label y_i and morphological tag z, the suffix, deletion rule, and case are (f, d, C). (5) According to label y_i and morphological tag z, the suffix, deletion rule, and number are (f, d, N).

We train a set of linear weights on our features using the averaged structured perceptron algorithm (Collins, 2002).

4 Experiments

In this section we turn to experimental findings to provide empirical support for our proposed framework.

Corpus: To test our cross-lingual model, we apply it to a morphologically analyzed corpus of eight languages (Erjavec, 2004). The corpus includes a roughly 100,000 word English text, Orwell's novel "Nineteen Eighty Four," and its translation into seven languages: Bulgarian, Czech, Estonian, Hungarian, Romanian, Slovene, and Serbian. All the words in the corpus are tagged with morphological stems and a detailed morpho-syntactic analysis. Although the texts are parallel, we note that parallelism is nowhere assumed nor exploited by our model. See Table 1 for a summary of relevant corpus statistics. As indicated in the table, the raw number of nominal word types varies quite a bit across the languages, almost doubling from 4,178 (English) to 8,051 (Hungarian). In contrast, the number of stems appearing within these words is relatively stable across languages, ranging from a minimum of 3,112 (Bulgarian) to a maximum of 3,746 (Hungarian), an increase of just 20%.

In contrast, the number of suffixes across the languages varies quite a bit. Hungarian and Estonian, both Uralic languages with very complex nominal morphology, use 231 and 141 nominal suffixes, respectively. Besides English, the remaining languages employ between 21 and 32 suffixes, and English is the outlier in the other direction, with just three nominal inflectional suffixes.

Baselines and Results: As our unsupervised monolingual baseline, we use the Linguistica program (Goldsmith, 2001; Goldsmith, 2005). We apply Linguistica's default settings, and run the "suffix prediction" option. Our model's search procedure closely mirrors the one used by Linguistica, with the crucial difference that instead of attempting to greedily minimize description length, our algorithm instead tries to find the analysis as close as possible in the universal feature space to that of another language.

To apply our model, we treat each of the eight

	Linguistica	Our Model						Supervised
		Nearest Neighbor		Self (oracle)		Avg.		
		Accuracy	Distance	Accuracy	Distance	Accuracy	Distance	
BG	68.7	84.0 (RO)	0.13	88.7	0.03	68.6	3.90	94.7
CS	60.4	82.8 (BG)	0.40	84.5	0.03	66.3	4.05	93.5
EN	81.1	75.8 (BG)	1.29	89.3	0.10	58.3	4.30	93.4
ET	51.2	66.6 (HU)	0.35	80.9	0.03	52.8	4.57	86.5
HU	64.5	69.3 (ET)	0.81	66.5	1.10	68.0	4.94	94.9
RO	65.6	71.0 (CS)	0.11	71.2	0.15	62.3	3.95	89.1
SL	61.1	82.8 (SR)	0.07	85.5	0.04	61.7	3.69	95.4
SR	64.2	79.1 (SL)	0.06	82.2	0.04	63.0	3.71	94.8
avg.	64.6	76.4	0.40	81.1	0.19	62.6	4.14	92.8

Table 2: **Prediction accuracy** over word types for the Linguistica baseline, our cross-lingual model, and the monolingual supervised perceptron model. For our model, we provide both prediction accuracy and resulting distance to the training language in three different scenarios: (i) **Nearest Neighbor:** The training languages include all seven other languages in our data set, and the predictions with minimal distance to a training language are chosen (the nearest neighbor is indicated in parentheses). (ii) **Self (oracle):** Each language is trained to minimize the distance to its *own* gold-standard analysis. (iii) **Average:** The feature values of all seven training languages are averaged together to create a single objective.

languages in turn as the test language, with the other seven serving as training examples. For each test language, we iterate the search procedure for each training language (performed in parallel), until convergence. The number of required iterations varies from 6 to 36 (depending on the test-training language pair), and each iteration takes no more than 30 seconds of run-time on a 2.4GHz Intel Xeon E5620 processor. We also consider two variants of our method. In the first (**Self (oracle)**), we train each test language to minimize the distance to its *own* gold standard feature values. In the second variant (**Avg.**), we average the feature values of all seven training languages into a single objective. As a plausible upper bound on performance, we implemented the structured perceptron described in Section 3.2. For each language, we train the perceptron on a randomly selected set of 80% of the nouns, and test on the remaining 20%.

The prediction accuracy for all models is calculated as the fraction of word types with correctly predicted suffixes. See Table 2 for the results. For all languages other than English (which is a morphological loner in our group of languages), our model improves over the baseline by a substantial margin, yielding an average increase of 11.8 absolute percentage points, and a reduction in error relative to the supervised upper bound of 42%. Some of the most striking improvements are seen on Serbian and Slovene. These languages are closely related to one another, and indeed our model discovers that they are each others' nearest neighbors. By guiding their morphological analyses towards one another, our model achieves a 21 percentage point increase in the case of Slovene and a 15 percentage point increase in the case of Slovene.

Perhaps unsurprisingly, when each language's gold standard feature values are used as its *own* target (**Self (oracle)** in Table 2), performance increases even further, to an average of 81.1%. By the same token, the resulting distance in universal feature space between training and test analyses is cut in half under this variant, when compared to the non-oracular nearest neighbor method. The remaining errors may be due to limitations of the search procedure (i.e. getting caught in local minima), or to the coarseness of the feature space (i.e. incorrect analyses might map to the same feature values as the correct analysis). Finally, we note that minimizing the distance to the *average* feature values of the seven training languages (**Avg.** in Table 2) yields subpar performance and very large distances between between predicted analyses and target feature values (4.14 compared to 0.40 for nearest neighbor). This

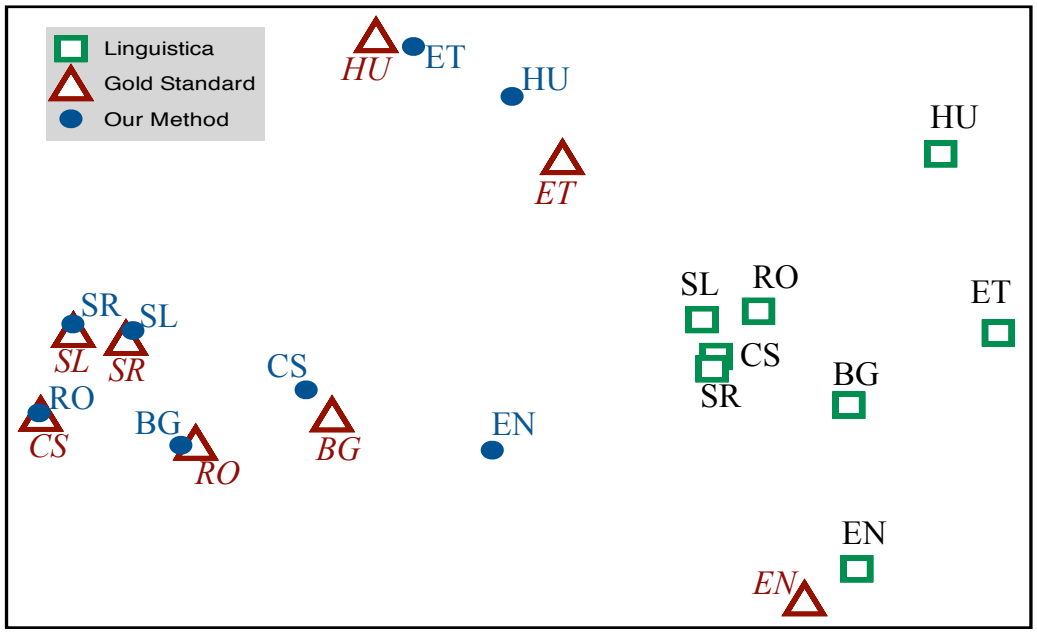


Figure 2: **Locations in Feature Space** of Linguistica predictions (green squares), gold standard analyses (red triangles), and our model's nearest neighbor predictions (blue circles). The original 8-dimensional feature space was reduced to two dimensions using Multidimensional Scaling.

result may indicate that the average feature point between training languages is simply unattainable as an analysis of a real lexicon of nouns.

Visualizing Locations in Feature Space: Besides assessing our method quantitatively, we can also visualize the the eight languages in universal feature space according to (i) their gold standard analyses, (ii) the predictions of our model and (iii) the predictions of Linguistica. To do so, we reduce the 8-dimensional features space down to two dimensions while preserving the distances between the predicted and gold standard feature vectors, using Multidimensional Scaling (MDS). The results of this analysis are shown in Figure 2. With the exception of English, our model's analyses lie closer in feature space to their gold standard counterparts than those of the baseline. It is interesting to note that Serbian and Slovene, which are very similar languages, have essentially swapped places under our model's analysis, as have Estonian and Hungarian (both highly inflected Uralic languages). English has (unfortunately) been pulled towards Bulgarian, the second least inflecting language in our set.

Learning Curves: We also measured the performance of our method as a function of the number of languages in the training set. For each target language, we consider all possible training sets of sizes ranging from 1 to 7 and select the predictions which bring our test language closest in distance to one of the languages in the set. We then average the resulting accuracy over all training sets of each size. Figure 3 shows the resulting learning curves averaged over all test languages (left), as well as broken down by test language (right). The overall trend is clear: as additional languages are added to the training set, test performance improves. In fact, with only one training language, our method performs worse (on average) than the Linguistica baseline. However, with two or more training languages available, our method achieves superior results.

Accuracy vs. Distance: We can gain some insight into these learning curves if we consider the relationship between accuracy (of the test language analysis) and distance to the training language (of the same predicted analysis). The more training languages available, the greater the chance that we can guide our test language into very close proximity to

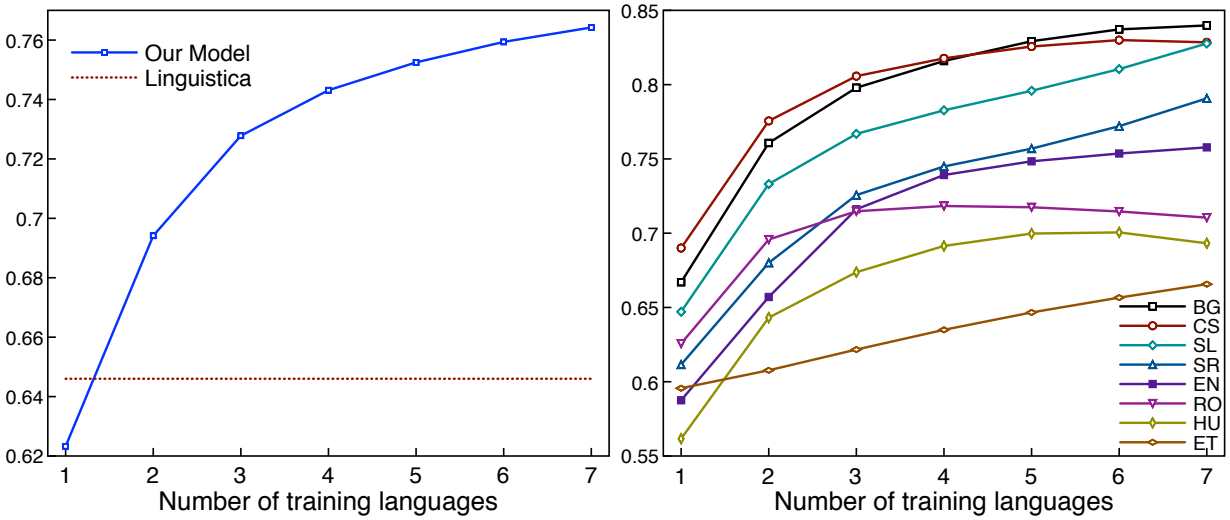


Figure 3: **Learning curves** for our model as the number of training languages increases. The figure on the left shows the average accuracy of all eight languages for increasingly larger training sets (results are averaged over all training sets of size 1,2,3,...). The dotted line indicates the average performance of the baseline. The figure on the right shows similar learning curves, broken down individually for each test language (see Figure 1 for language abbreviations).

one of them. It thus stands to reason that a strong (negative) correlation between distance and accuracy would lead to increased accuracy with larger training sets. In order to assess this correlation, we considered all 56 test-train language pairs and collected the resulting accuracy and distance for each pair. We separately scaled accuracy and distance to the unit interval for each test language (as some test languages are inherently more difficult than others). The resulting plot, shown in Figure 4, shows the expected correlation: When our test language can be guided very closely to the training language, the resulting predictions are likely to be good. If not, the predictions are likely to be bad.

5 Conclusions and Future Work

The approach presented in this paper recasts morphological induction as a structured prediction task. We assume the presence of morphologically labeled languages as *training examples* which guide the induction process for unlabeled test languages. We developed a novel structured nearest neighbor approach for this task, in which all languages and their morphological analyses lie in a universal feature space. The task of the learner is to search through the space of morphological analyses for the test language and return the result which lies closest to one

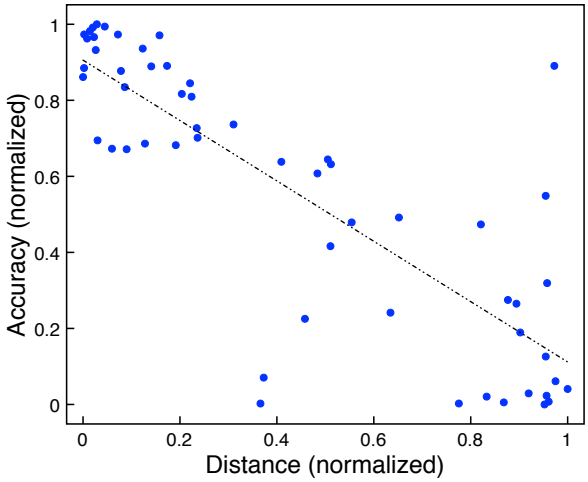


Figure 4: **Accuracy vs. Distance:** For all 56 possible test-train language pairs, we computed test accuracy along with resulting distance in universal feature space to the training language. Distance and accuracy are separately normalized to the unit interval for each test language, and all resulting points are plotted together. A line is fit to the points using least-squares regression.

of the training languages. Our empirical findings validate this approach: On a set of eight different languages, our method yields substantial accuracy gains over a traditional MDL-based approach in the task of nominal morphological induction.

One possible shortcoming of our approach is that it assumes a uniform weighting of the cross-lingual feature space. In fact, some features may be far more relevant than others in guiding our test language to an accurate analysis. In future work, we plan to integrate distance metric learning into our approach, allowing some features to be weighted more heavily than others. Besides potential gains in prediction accuracy, this approach may shed light on deeper relationships between languages than are otherwise apparent.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the ACL/CONLL*, pages 665–672.

Emily M. Bender. 2009. Linguistically naïve != language independent: why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics*, pages 26–32, Morristown, NJ, USA. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *Proceedings of the ACL*, pages 1288–1297, Uppsala, Sweden, July. Association for Computational Linguistics.

P. Blunsom, T. Cohn, and M. Osborne. 2009. Bayesian synchronous grammar induction. *Advances in Neural Information Processing Systems*, 21:161–168.

David Burkett, Slav Petrov, John Blitzer, and Dan Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *Proceedings of CoNLL*.

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of the NAACL/HLT*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.

T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0. Publications in Computer and Information Science Report A81, Helsinki University of Technology.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1).

Ido Dagan, Alon Itai, and Ulrike Schwall. 1991. Two languages are more informative than one. In *Proceedings of the ACL*, pages 130–137.

Sajib Dasgupta and Vincent Ng. 2007. Unsupervised part-of-speech acquisition for resource-scarce languages. In *Proceedings of the EMNLP-CoNLL*, pages 218–227.

T. Erjavec. 2004. MULTEXT-East version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *Fourth International Conference on Language Resources and Evaluation, LREC*, volume 4, pages 1535–1538.

John Goldsmith. 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 27(2):153–198.

John Goldsmith. 2005. An algorithm for the unsupervised learning of morphology. Technical report, University of Chicago.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11(3):311–325.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. 2009. Multilingual part-of-speech tagging: two unsupervised approaches. *Journal of Artificial Intelligence Research*, 36(1):341–385.

Sebastian Padó and Mirella Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of ACL*, pages 1161 – 1168.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 209–217, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philip Resnik and David Yarowsky. 1997. A perspective on word sense disambiguation methods and their evaluation. In *Proceedings of the ACL SIGLEX Workshop*

on Tagging Text with Lexical Semantics: Why, What, and How?, pages 79–86.

Patrick Schone and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–9, Morristown, NJ, USA. Association for Computational Linguistics.

Benjamin Snyder and Regina Barzilay. 2008a. Cross-lingual propagation for morphological analysis. In *Proceedings of the AAAI*, pages 848–854.

Benjamin Snyder and Regina Barzilay. 2008b. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of the ACL/HLT*, pages 737–745.

Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. 2008. Unsupervised multilingual learning for POS tagging. In *Proceedings of EMNLP*, pages 1041–1050.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009a. Unsupervised multilingual grammar induction. In *Proceedings of the ACL*, pages 73–81.

Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. 2009b. Adding more languages improves unsupervised multilingual part-of-speech tagging: a Bayesian non-parametric approach. In *Proceedings of the NAACL*, pages 83–91.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the NAACL*, pages 1–8.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216, Morristown, NJ, USA. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2000. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*, pages 161–168.

Training a Log-Linear Parser with Loss Functions via Softmax-Margin

Michael Auli
School of Informatics
University of Edinburgh
m.auli@sms.ed.ac.uk

Adam Lopez
HLTCOE
Johns Hopkins University
alopez@cs.jhu.edu

Abstract

Log-linear parsing models are often trained by optimizing likelihood, but we would prefer to optimise for a task-specific metric like F-measure. Softmax-margin is a convex objective for such models that minimises a bound on expected risk for a given loss function, but its naïve application requires the loss to decompose over the predicted structure, which is not true of F-measure. We use softmax-margin to optimise a log-linear CCG parser for a variety of loss functions, and demonstrate a novel dynamic programming algorithm that enables us to use it with F-measure, leading to substantial gains in accuracy on CCGBank. When we embed our loss-trained parser into a larger model that includes supertagging features incorporated via belief propagation, we obtain further improvements and achieve a labelled/unlabelled dependency F-measure of 89.3%/94.0% on gold part-of-speech tags, and 87.2%/92.8% on automatic part-of-speech tags, the best reported results for this task.

1 Introduction

Parsing models based on Conditional Random Fields (CRFs; Lafferty et al., 2001) have been very successful (Clark and Curran, 2007; Finkel et al., 2008). In practice, they are usually trained by maximising the conditional log-likelihood (CLL) of the training data. However, it is widely appreciated that optimizing for task-specific metrics often leads to better performance on those tasks (Goodman, 1996; Och, 2003).

An especially attractive means of accomplishing this for CRFs is the softmax-margin (SMM) objective (Sha and Saul, 2006; Povey and Woodland, 2008; Gimpel and Smith, 2010a) (§2). In addition to retaining a probabilistic interpretation and optimizing towards a loss function, it is also convex, making it straightforward to optimise. Gimpel and Smith (2010a) show that it can be easily implemented with a simple change to standard likelihood-based training, provided that the loss function decomposes over the predicted structure.

Unfortunately, the widely-used F-measure metric does not decompose over parses. To solve this, we introduce a novel dynamic programming algorithm that enables us to compute the exact quantities needed under the softmax-margin objective using F-measure as a loss (§3). We experiment with this and several other metrics, including precision, recall, and decomposable approximations thereof. Our ability to optimise towards exact metrics enables us to verify the effectiveness of more efficient approximations. We test the training procedures on the state-of-the-art Combinatory Categorial Grammar (CCG; Steedman 2000) parser of Clark and Curran (2007), obtaining substantial improvements under a variety of conditions. We then embed this model into a more accurate model that incorporates additional supertagging features via loopy belief propagation. The improvements are additive, obtaining the best reported results on this task (§4).

2 Softmax-Margin Training

The softmax-margin objective modifies the standard likelihood objective for CRF training by reweighting

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 333–343,
Edinburgh, Scotland, UK, July 27–31, 2011.

each possible outcome of a training input according to its *risk*, which is simply the loss incurred on a particular example. This is done by incorporating the loss function directly into the linear scoring function of an individual example.

Formally, we are given m training pairs $(x^{(1)}, y^{(1)})...(x^{(m)}, y^{(m)})$, where each $x^{(i)} \in \mathcal{X}$ is drawn from the set of possible inputs, and each $y^{(i)} \in \mathcal{Y}(x^{(i)})$ is drawn from a set of possible instance-specific outputs. We want to learn the K parameters θ of a log-linear model, where each $\lambda_k \in \theta$ is the weight of an associated feature $h_k(x, y)$. Function $f(x, y)$ maps input/output pairs to the vector $h_1(x, y)...h_K(x, y)$, and our log-linear model assigns probabilities in the usual way.

$$p(y|x) = \frac{\exp\{\theta^\mathsf{T} f(x, y)\}}{\sum_{y' \in \mathcal{Y}(x)} \exp\{\theta^\mathsf{T} f(x, y')\}} \quad (1)$$

The conditional log-likelihood objective function is given by Eq. 2 (Figure 1). Now consider a function $\ell(y, y')$ that returns the loss incurred by choosing to output y' when the correct output is y. The softmax-margin objective simply modifies the unnormalised, unexponentiated score $\theta^\mathsf{T} f(x, y')$ by adding $\ell(y, y')$ to it. This yields the objective function (Eq. 3) and gradient computation (Eq. 4) shown in Figure 1.

This straightforward extension has several desirable properties. In addition to having a probabilistic interpretation, it is related to maximum margin and minimum-risk frameworks, it can be shown to minimise a bound on expected risk, and it is convex (Gimpel and Smith, 2010b).

We can also see from Eq. 4 that the only difference from standard CLL training is that we must compute feature expectations with respect to the cost-augmented scoring function. As Gimpel and Smith (2010a) discuss, if the loss function decomposes over the predicted structure, we can treat its decomposed elements as unweighted features that fire on the corresponding structures, and compute expectations in the normal way. In the case of our parser, where we compute expectations using the inside-outside algorithm, a loss function decomposes if it decomposes over spans or productions of a CKY chart.

3 Loss Functions for Parsing

Ideally, we would like to optimise our parser towards a task-based evaluation. Our CCG parser is evaluated on labeled, directed dependency recovery using F-measure (Clark and Hockenmaier, 2002). Under this evaluation we will represent output y' and ground truth y as variable-sized sets of dependencies. We can then compute precision $P(y, y')$, recall $R(y, y')$, and F-measure $F_1(y, y')$.

$$P(y, y') = \frac{|y \cap y'|}{|y'|} \quad (5)$$

$$R(y, y') = \frac{|y \cap y'|}{|y|} \quad (6)$$

$$F_1(y, y') = \frac{2PR}{P + R} = \frac{2|y \cap y'|}{|y| + |y'|} \quad (7)$$

These metrics are positively correlated with performance – they are *gain* functions. To incorporate them in the softmax-margin framework we reformulate them as loss functions by subtracting from one.

3.1 Computing F-Measure-Augmented Expectations at the Sentence Level

Unfortunately, none of these metrics decompose over parses. However, the individual statistics that are used to compute them *do* decompose, a fact we will exploit to devise an algorithm that computes the necessary expectations. Note that since y is fixed, F_1 is a function of two integers: $|y \cap y'|$, representing the number of correct dependencies in y'; and $|y'|$, representing the total number of dependencies in y', which we will denote as n and d, respectively.[1] Each pair $\langle n, d \rangle$ leads to a different value of F_1. Importantly, both n and d decompose over parses.

The key idea will be to treat F_1 as a non-local feature of the parse, dependent on values n and d.[2] To compute expectations we split each span in an otherwise usual inside-outside computation by all pairs $\langle n, d \rangle$ incident at that span.

Formally, our goal will be to compute expectations over the sentence $a_1...a_L$. In order to abstract away from the particulars of CCG we present the algorithm in relatively familiar terms as a variant of

[1] For *numerator* and *denominator*.

[2] This is essentially the same trick used in the oracle F-measure algorithm of Huang (2008), and indeed our algorithm is a sum-product variant of that max-product algorithm.

$$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y)\} \right] \quad (2)$$

$$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\mathsf{T}} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y) + \ell(y^{(i)}, y)\} \right] \quad (3)$$

$$\frac{\partial}{\partial \lambda_k} = \sum_{i=1}^{m} \left[-h_k(x^{(i)}, y^{(i)}) + \sum_{y \in \mathcal{Y}(x^{(i)})} \frac{\exp\{\theta^{\mathsf{T}} f(x^{(i)}, y) + \ell(y^{(i)}, y)\}}{\sum_{y' \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\mathsf{T}} f(x^{(i)}, y') + \ell(y^{(i)}, y')\}} h_k(x^{(i)}, y) \right] \quad (4)$$

Figure 1: Conditional log-likelihood (Eq. 2), Softmax-margin objective (Eq. 3) and gradient (Eq. 4).

the classic inside-outside algorithm (Baker, 1979). We use the notation $a : A$ for lexical entries and $BC \Rightarrow A$ to indicate that categories B and C combine to form category A via forward or backward composition or application.[3] The weight of a rule is denoted with w. The classic algorithm associates inside score $I(A_{i,j})$ and outside score $O(A_{i,j})$ with category A spanning sentence positions i through j, computed via the following recursions.

$$\begin{aligned}
I(A_{i,i+1}) &= w(a_{i+1} : A) \\
I(A_{i,j}) &= \sum_{k,B,C} I(B_{i,k}) I(C_{k,j}) w(BC \Rightarrow A) \\
I(GOAL) &= I(S_{0,L}) \\
O(GOAL) &= 1 \\
O(A_{i,j}) &= \sum_{k,B,C} O(C_{i,k}) I(B_{j,k}) w(AB \Rightarrow C) + \\
&\quad \sum_{k,B,C} O(C_{k,j}) I(B_{k,i}) w(BA \Rightarrow C)
\end{aligned}$$

The expectation of A spanning positions i through j is then $I(A_{i,j})O(A_{i,j})/I(GOAL)$.

Our algorithm extends these computations to state-split items $A_{i,j,n,d}$.[4] Using functions $n_+(\cdot)$ and $d_+(\cdot)$ to respectively represent the number of correct and total dependencies introduced by a parsing action, we present our algorithm in Fig. 3. The final inside equation and initial outside equation incorporate the loss function for all derivations having a particular F-score, enabling us to obtain the desired expectations. A simple modification of the goal equations enables us to optimise precision, recall or a weighted F-measure.

To analyze the complexity of this algorithm, we must ask: how many pairs $\langle n, d \rangle$ can be incident at each span? A CCG parser does not necessarily return one dependency per word (see Figure 2 for an example), so d is not necessarily equal to the sentence length L as it might be in many dependency parsers, though it is still bounded by $\mathcal{O}(L)$. However, this behavior is sufficiently uncommon that we expect all parses of a sentence, good or bad, to have close to L dependencies, and hence we expect the range of d to be constant on average. Furthermore, n will be bounded from below by zero and from above by $\min(|y|, |y'|)$. Hence the set of all possible F-measures for all possible parses is bounded by $\mathcal{O}(L^2)$, but on average it should be closer to $\mathcal{O}(L)$. Following McAllester (1999), we can see from inspection of the free variables in Fig. 3 that the algorithm requires worst-case $\mathcal{O}(L^7)$ and average-case $\mathcal{O}(L^5)$ time complexity, and worse-case $\mathcal{O}(L^4)$ and average-case $\mathcal{O}(L^3)$ space complexity.

Note finally that while this algorithm computes exact sentence-level expectations, it is approximate at the corpus level, since F-measure does not decompose over sentences. We give the extension to exact corpus-level expectations in Appendix A.

3.2 Approximate Loss Functions

We will also consider approximate but more efficient alternatives to our exact algorithms. The idea is to use cost functions which only utilise statistics

[3] These correspond respectively to unary rules $A \rightarrow a$ and binary rules $A \rightarrow BC$ in a Chomsky normal form grammar.

[4] Here we use *state-splitting* to refer to splitting an item $A_{i,j}$ into many items $A_{i,j,n,d}$, one for each $\langle n, d \rangle$ pair.

$$I(A_{i,i+1,n,d}) = w(a_{i+1} : A) \quad \text{iff} n = n_+(a_{i+1} : A), d = d_+(a_{i+1} : A)$$

$$I(A_{i,j,n,d}) = \sum_{k,B,C} \sum_{\substack{\{n',n'':n'+n''+n_+(BC \Rightarrow A)=n\},\\ \{d',d'':d'+d''+d_+(BC \Rightarrow A)=d\}}} I(B_{i,k,n',d'})I(C_{k,j,n'',d''})w(BC \Rightarrow A)$$

$$I(GOAL) = \sum_{n,d} I(S_{0,L,n,d}) \left(1 - \frac{2n}{d + |y|}\right)$$

$$O(S_{0,N,n,d}) = \left(1 - \frac{2n}{d + |y|}\right)$$

$$O(A_{i,j,n,d}) = \sum_{k,B,C} \sum_{\substack{\{n',n'':n'-n''-n_+(AB \Rightarrow C)=n\},\\ \{d',d'':d'-d''-d_+(AB \Rightarrow C)=d\}}} O(C_{i,k,n',d'})I(B_{j,k,n'',d''})w(AB \Rightarrow C) +$$

$$\sum_{k,B,C} \sum_{\substack{\{n',n'':n'-n''-n_+(BA \Rightarrow C)=n\},\\ \{d',d'':d'-d''-d_+(BA \Rightarrow C)=d\}}} O(C_{k,j,n',d'})I(B_{k,i,n'',d''})w(BA \Rightarrow C)$$

Figure 3: State-split inside and outside recursions for computing softmax-margin with F-measure.

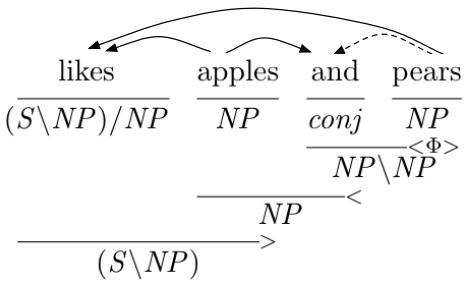


Figure 2: Example of flexible dependency realisation in CCG: Our parser (Clark and Curran, 2007) creates dependencies arising from coordination once all conjuncts are found and treats "and" as the syntactic head of coordinations. The coordination rule (Φ) does not yet establish the dependency "and - pears" (dotted line); it is the backward application ($<$) in the larger span, "apples and pears", that establishes it, together with "and - pears". CCG also deals with unbounded dependencies which potentially lead to more dependencies than words (Steedman, 2000); in this example a unification mechanism creates the dependencies "likes - apples" and "likes - pears" in the forward application ($>$). For further examples and a more detailed explanation of the mechanism as used in the C&C parser refer to Clark et al. (2002).

available within the current local structure, similar to those used by Taskar et al. (2004) for tracking constituent errors in a context-free parser. We design three simple losses to approximate precision, recall and F-measure on CCG dependency structures.

Let $T(y)$ be the set of parsing actions required to build parse y. Our decomposable approximation to precision simply counts the number of incorrect dependencies using the local dependency counts, $n_+(\cdot)$ and $d_+(\cdot)$.

$$DecP(y) = \sum_{t \in T(y)} d_+(t) - n_+(t) \qquad (8)$$

To compute our approximation to recall we require the number of gold dependencies, $c_+(\cdot)$, which should have been introduced by a particular parsing action. A gold dependency is due to be recovered by a parsing action if its head lies within one child span and its dependent within the other. This yields a decomposed approximation to recall that counts the number of missed dependencies.

$$DecR(y) = \sum_{t \in T(y)} c_+(t) - n_+(t) \qquad (9)$$

Unfortunately, the flexible handling of dependencies in CCG complicates our formulation of c_+, rendering it slightly more approximate. The unification mechanism of CCG sometimes causes dependencies to be realised later in the derivation, at a point when both the head and the dependent are in the same span, violating the assumption used to compute c_+ (see again Figure 2). Exceptions like this can cause mismatches between n_+ and c_+. We set $c_+ = n_+$ whenever $c_+ < n_+$ to account for these occasional discrepancies.

Finally, we obtain a decomposable approximation to F-measure.

$$DecF1(y) = DecP(y) + DecR(y) \qquad (10)$$

4 Experiments

Parsing Strategy. CCG parsers use a pipeline strategy: we first multitag each word of the sentence with a small subset of its possible lexical categories using a *supertagger*, a sequence model over these categories (Bangalore and Joshi, 1999; Clark, 2002). Then we parse the sentence under the requirement that the lexical categories are fixed to those preferred by the supertagger. In our experiments we used two variants on this strategy.

First is the *adaptive supertagging* (AST) approach of Clark and Curran (2004). It is based on a step function over supertagger beam widths, relaxing the pruning threshold for lexical categories only if the parser fails to find an analysis. The process either succeeds and returns a parse after some iteration or gives up after a predefined number of iterations. As Clark and Curran (2004) show, most sentences can be parsed with very tight beams.

Reverse adaptive supertagging is a much less aggressive method that seeks only to make sentences parsable when they otherwise would not be due to an impractically large search space. Reverse AST starts with a wide beam, narrowing it at each iteration only if a maximum chart size is exceeded. Table 1 shows beam settings for both strategies.

Adaptive supertagging aims for speed via pruning while the reverse strategy aims for accuracy by exposing the parser to a larger search space. Although Clark and Curran (2007) found no actual improvements from the latter strategy, we will show that with our softmax-margin-trained models it can have a substantial effect.

Parser. We use the C&C parser (Clark and Curran, 2007) and its supertagger (Clark, 2002). Our baseline is the hybrid model of Clark and Curran (2007), which contains features over both normal-form derivations and CCG dependencies. The parser relies solely on the supertagger for pruning, using exact CKY for search over the pruned space. Training requires calculation of feature expectations over packed charts of derivations. For training, we limited the number of items in this chart to 0.3 million, and for testing, 1 million. We also used a more permissive training supertagger beam (Table 2) than in previous work (Clark and Curran, 2007). Models were trained with the parser's L-BFGS trainer.

Evaluation. We evaluated on CCGbank (Hockenmaier and Steedman, 2007), a right-most normal-form CCG version of the Penn Treebank. We use sections 02-21 (39603 sentences) for training, section 00 (1913 sentences) for development and section 23 (2407 sentences) for testing. We supply gold-standard part-of-speech tags to the parsers. We evaluate on labelled and unlabelled predicate argument structure recovery and supertag accuracy.

4.1 Training with Maximum F-measure Parses

So far we discussed how to optimise towards task-specific metrics via changing the training objective. In our first experiment we change the *data* on which we optimise CLL. This is a kind of simple baseline to our later experiments, attempting to achieve the same effect by simpler means. Specifically, we use the algorithm of Huang (2008) to generate oracle F-measure parses for each sentence. Updating towards these oracle parses corrects the reachability problem in standard CLL training. Since the supertagger is used to prune the training forests, the correct parse is sometimes pruned away – reducing data utilisation to 91%. Clark and Curran (2007) correct for this by adding the gold tags to the parser input. While this increases data utilisation, it biases the model by training in an idealised setting not available at test time. Using oracle parses corrects this bias while permitting 99% data utilisation. The labelled F-score of the oracle parses lies at 98.1%. Though we expected that this might result in some improvement, results (Table 3) show that this has no

Condition	Parameter	Iteration 1	2	3	4	5
`AST`	β (beam width)	0.075	0.03	0.01	0.005	0.001
	k (dictionary cutoff)	20	20	20	20	150
`Reverse`	β	0.001	0.005	0.01	0.03	0.075
	k	150	20	20	20	20

Table 1: Beam step function used for standard (`AST`) and less aggressive (`Reverse`) AST throughout our experiments. Parameter β is a beam threshold while k bounds the number of lexical categories considered for each word.

Condition	Parameter	Iteration 1	2	3	4	5	6	7
`Training`	β	0.001	0.001	0.0045	0.0055	0.01	0.05	0.1
	k	150	20	20	20	20	20	20
`C&C '07`	β	0.0045	0.0055	0.01	0.05	0.1		
	k	20	20	20	20	20		

Table 2: Beam step functions used for training: The first row shows the large scale settings used for most experiments and the standard C&C settings. (cf. Table 1)

	LF	LP	LR	UF	UP	UR	Data Util (%)
Baseline	87.40	87.85	86.95	93.11	93.59	92.63	91%
Max-F Parses	87.46	87.95	86.98	93.09	93.61	92.57	99%
CCGbank+Max-F	87.45	87.96	86.94	93.09	93.63	92.55	99%

Table 3: Performance on section 00 of CCGbank when comparing models trained with treebank-parses (Baseline) and maximum F-score parses (Max-F) using adaptive supertagging as well as a combination of CCGbank and Max-F parses. Evaluation is based on labelled and unlabelled F-measure (LF/UF), precision (LP/UP) and recall (LR/UR).

effect. However, it does serve as a useful baseline.

4.2 Training with the Exact Algorithm

We first tested our assumptions about the feasibility of training with our exact algorithm by measuring the amount of state-splitting. Figure 4 plots the average number of splits per span against the relative span-frequency; this is based on a typical set of training forests containing over 600 million states. The number of splits increases exponentially with span size but equally so decreases the number of spans with many splits. Hence the small number of states with a high number of splits is balanced by a large number of spans with only a few splits: The highest number of splits per span observed with our settings was 4888 but we find that the average number of splits lies at 44. Encouragingly, this enables experimentation in all but very large scale settings.

Figure 5 shows the distribution of n and d pairs across all split-states in the training corpus; since

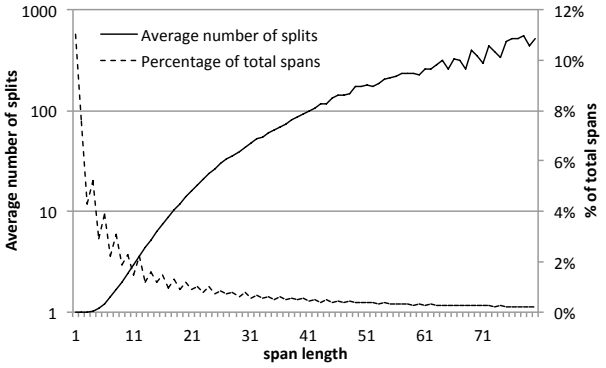


Figure 4: Average number of state-splits per span length as introduced by a sentence-level F-measure loss function. The statistics are averaged over the training forests generated using the settings described in §4.

n, the number of correct dependencies, over d, the number of all recovered dependencies, is precision, the graph shows that only a minority of states have either very high or very low precision. The range of values suggests that the softmax-margin criterion

will have an opportunity to substantially modify the expectations, hopefully to good effect.

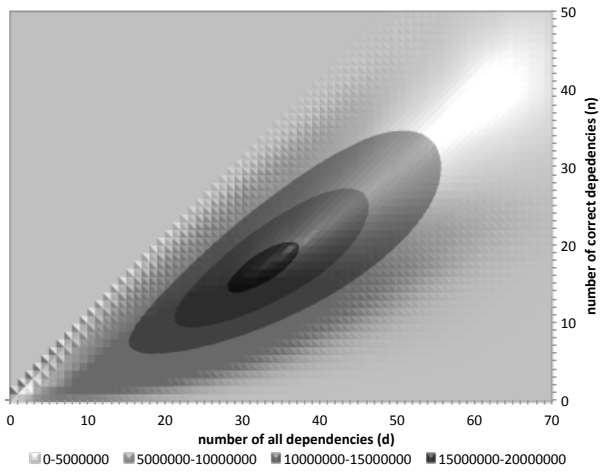


Figure 5: Distribution of states with d dependencies of which n are correct in the training forests.

We next turn to the question of optimization with these algorithms. Due to the significant computational requirements, we used the computationally less intensive normal-form model of Clark and Curran (2007) as well as their more restrictive training beam settings (Table 2). We train on all sentences of the training set as above and test with AST.

In order to provide greater control over the influence of the loss function, we introduce a multiplier τ, which simply amends the second term of the objective function (3) to:

$$\log \sum_{y \in Y(x^i)} \exp\{\theta^T f(x^i, y) + \tau \times \ell(y^i, y)\}$$

Figure 6 plots performance of the exact loss functions across different settings of τ on various evaluation criteria, for models restricted to at most 3000 items per chart at training time to allow rapid experimentation with a wide parameter set. Even in this constrained setting, it is encouraging to see that each loss function performs best on the criteria it optimises. The precision-trained parser also does very well on F-measure; this is because the parser has a tendency to perform better in terms of precision than recall.

4.3 Exact vs. Approximate Loss Functions

With these results in mind, we conducted a comparison of parsers trained using our exact and approximate loss functions. Table 4 compares their performance head to head when restricting training chart sizes to 100,000 items per sentence, the largest setting our computing resources allowed us to experiment with. The results confirm that the loss-trained models improve over a likelihood-trained baseline, and furthermore that the exact loss functions seem to have the best performance. However, the approximations are extremely competitive with their exact counterparts. Because they are also efficient, this makes them attractive for larger-scale experiments. Training time increases by an order of magnitude with exact loss functions despite increased theoretical complexity (§3.1); there is no significant change with approximate loss functions.

Table 5 shows performance of the approximate losses with the large scale settings initially outlined (§4). One striking result is that the softmax-margin trained models coax more accurate parses from the larger search space, in contrast to the likelihood-trained models. Our best loss model improves the labelled F-measure by over 0.8%.

4.4 Combination with Integrated Parsing and Supertagging

As a final experiment, we embed our loss-trained model into an integrated model that incorporates Markov features over supertags into the parsing model (Auli and Lopez, 2011). These features have serious implications on search: even allowing for the observation of Fowler and Penn (2010) that our CCG is weakly context-free, the search problem is equivalent to finding the optimal derivation in the weighted intersection of a regular and context-free language (Bar-Hillel et al., 1964), making search very expensive. Therefore parsing with this model requires approximations.

To experiment with this combined model we use loopy belief propagation (LBP; Pearl et al., 1985), previously applied to dependency parsing by Smith and Eisner (2008). A more detailed account of its application to our combined model can be found in (2011), but we sketch the idea here. We construct a graphical model with two factors: one is a distribu-

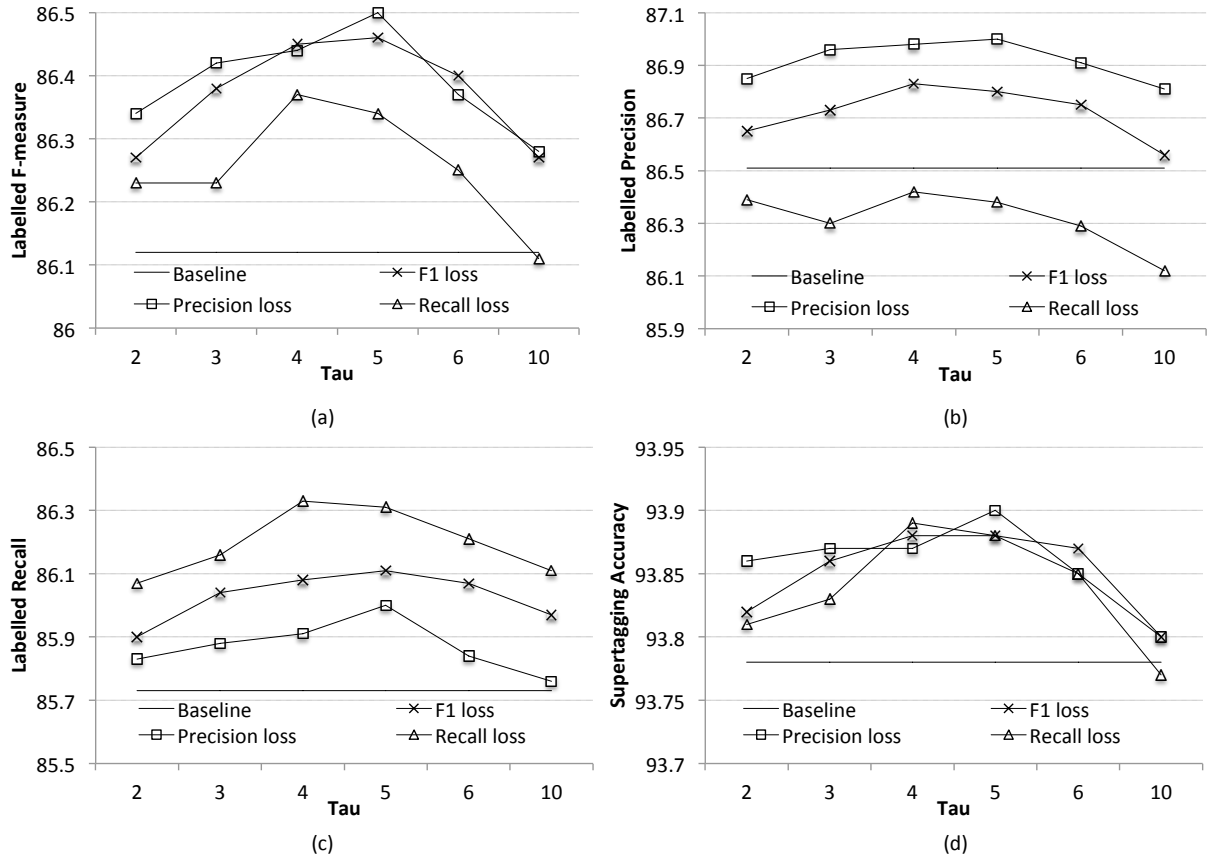


Figure 6: Performance of exact cost functions optimizing F-measure, precision and recall in terms of (a) labelled F-measure, (b) precision, (c) recall and (d) supertag accuracy across various settings of τ on the development set.

	section 00 (dev)						section 23 (test)					
	LF	LP	LR	UF	UP	UR	LF	LP	LR	UF	UP	UR
CLL	86.76	87.16	86.36	92.73	93.16	92.30	87.46	87.80	87.12	92.85	93.22	92.49
DecP	87.18	**87.93**	86.44	92.93	93.73	92.14	87.75	**88.34**	87.17	93.04	**93.66**	92.43
DecR	87.31	87.55	87.07	93.00	93.26	92.75	87.57	87.71	87.42	92.92	93.07	92.76
DecF1	87.27	87.78	86.77	93.04	93.58	92.50	87.69	88.10	87.28	93.04	93.48	92.61
P	87.25	87.85	86.66	92.99	**93.63**	92.36	**87.76**	88.23	87.30	**93.06**	93.55	92.57
R	**87.34**	87.51	**87.16**	92.98	93.17	**92.80**	87.57	87.62	**87.51**	92.92	92.98	**92.86**
F1	**87.34**	87.74	86.94	**93.05**	93.47	92.62	87.71	88.01	87.41	93.02	93.34	92.70

Table 4: Performance of exact and approximate loss functions against conditional log-likelihood (CLL): decomposable precision (DecP), recall (DecR) and F-measure (DecF1) versus exact precision (P), recall (R) and F-measure (F1). Evaluation is based on labelled and unlabelled F-measure (LF/UF), precision (LP/UP) and recall (LR/UR).

	section 00 (dev)						section 23 (test)					
	AST			Reverse			AST			Reverse		
	LF	UF	ST	LF	UF	ST	LF	UF	ST	LF	UF	ST
CLL	87.38	93.08	94.21	87.36	93.13	93.99	87.73	93.09	94.33	87.65	93.06	94.01
DecP	87.35	92.99	94.25	87.75	93.25	94.22	**88.10**	93.26	94.51	88.51	93.50	94.39
DecR	87.48	93.00	94.34	87.70	93.16	94.30	87.66	92.83	94.38	87.77	92.91	94.22
DecF1	**87.67**	**93.23**	**94.39**	**88.12**	**93.52**	**94.46**	88.09	**93.28**	**94.50**	**88.58**	**93.57**	**94.53**

Table 5: Performance of decomposed loss functions in large-scale training setting. Evaluation is based on labelled and unlabelled F-measure (LF/UF) and supertag accuracy (ST).

tion over supertag variables defined by a supertagging model, and the other is a distribution over these variables and a set of span variables defined by our parsing model.[5] The factors communicate by passing messages across the shared supertag variables that correspond to their marginal distributions over those variables. Hence, to compute approximate expectations across the entire model, we run forward-backward to obtain posterior supertag assignments. These marginals are passed as inside values to the inside-outside algorithm, which returns a new set of posteriors. The new posteriors are incorporated into a new iteration of forward-backward, and the algorithm iterates until convergence, or until a fixed number of iterations is reached – we found that a single iteration is sufficient, corresponding to a truncated version of the algorithm in which posteriors are simply passed from the supertagger to the parser. To decode, we use the posteriors in a minimum-risk parsing algorithm (Goodman, 1996).

Our baseline models are trained separately as before and combined at test time. For softmax-margin, we combine a parsing model trained with F1 and a supertagger trained with Hamming loss. Table 6 shows the results: we observe a gain of up to 1.5% in labelled F1 and 0.9% in unlabelled F1 on the test set. The loss functions prove their robustness by improving the more accurate combined models up to 0.4% in labelled F1. Table 7 shows results with automatic part-of-speech tags and a direct comparison with the Petrov parser trained on CCGbank (Fowler and Penn, 2010) which we outpeform on all metrics.

[5]These complex factors resemble those of Smith and Eisner (2008) and Dreyer and Eisner (2009); they can be thought of as case-factor diagrams (McAllester et al., 2008)

5 Conclusion and Future Work

The softmax-margin criterion is a simple and effective approach to training log-linear parsers. We have shown that it is possible to compute exact sentence-level losses under standard parsing metrics, not only approximations (Taskar et al., 2004). This enables us to show the effectiveness of these approximations, and it turns out that they are excellent substitutes for exact loss functions. Indeed, the approximate losses are as easy to use as standard conditional log-likelihood.

Empirically, softmax-margin training improves parsing performance across the board, beating the state-of-the-art CCG parsing model of Clark and Curran (2007) by up to 0.8% labelled F-measure. It also proves robust, improving a stronger baseline based on a combined parsing and supertagging model. Our final result of 89.3%/94.0% labelled and unlabelled F-measure is the best result reported for CCG parsing accuracy, beating the original C&C baseline by up to 1.5%.

In future work we plan to scale our exact loss functions to larger settings and to explore training with loss functions within loopy belief propagation. Although we have focused on CCG parsing in this work, we expect our methods to be equally applicable to parsing with other grammar formalisms including context-free grammar or LTAG.

Acknowledgements

We would like to thank Stephen Clark, Christos Christodoulopoulos, Mark Granroth-Wilding, Gholamreza Haffari, Alexandre Klementiev, Tom Kwiatkowski, Kira Mourao, Matt Post, and Mark Steedman for helpful discussion related to this work and comments on previous drafts, and the

	section 00 (dev)						section 23 (test)					
	AST			Reverse			AST			Reverse		
	LF	UF	ST	LF	UF	ST	LF	UF	ST	LF	UF	ST
CLL	87.38	93.08	94.21	87.36	93.13	93.99	87.73	93.09	94.33	87.65	93.06	94.01
BP	87.67	93.26	94.43	88.35	93.72	94.73	88.25	93.33	94.60	88.86	93.75	94.84
+DecF1	**87.90**	**93.40**	**94.52**	**88.58**	**93.88**	**94.79**	88.32	93.32	94.66	89.15	93.89	94.98
+SA	87.73	93.28	94.49	88.40	93.71	94.75	**88.47**	**93.48**	**94.71**	**89.25**	**93.98**	**95.01**

Table 6: Performance of combined parsing and supertagging with belief propagation (BP); using decomposed-F1 as parser-loss function and supertag-accuracy (SA) as loss in the supertagger.

	section 00 (dev)						section 23 (test)					
	LF	LP	LR	UF	UP	UR	LF	LP	LR	UF	UP	UR
CLL	85.53	85.73	85.33	91.99	92.20	91.77	85.74	85.90	85.58	91.92	92.09	91.75
Petrov I-5	85.79	86.09	85.50	92.44	92.76	92.13	86.01	86.29	85.73	92.34	92.64	92.04
BP	86.45	86.75	86.17	92.60	92.92	92.29	86.84	87.08	86.61	92.57	92.82	92.32
+DecF1	**86.73**	**87.07**	**86.39**	**92.79**	**93.16**	**92.43**	87.08	87.37	86.78	92.68	93.00	92.37
+SA	86.51	86.86	86.16	92.60	92.98	92.23	**87.20**	**87.50**	**86.90**	**92.76**	**93.08**	**92.44**

Table 7: Results on automatically assigned POS tags. *Petrov I-5* is based on the parser output of Fowler and Penn (2010); evaluation is based on sentences for which all parsers returned an analysis.

anonymous reviewers for helpful comments. We also acknowledge funding from EPSRC grant EP/P504171/1 (Auli); and the resources provided by the Edinburgh Compute and Data Facility.

A Computing F-Measure-Augmented Expectations at the Corpus Level

To compute exact corpus-level expectations for softmax-margin using F-measure, we add an additional transition before reaching the $GOAL$ item in our original program. To reach it, we must parse every sentence in the corpus, associating statistics of aggregate $\langle n, d\rangle$ pairs for the entire training set in intermediate symbols $\Gamma^{(1)}...\Gamma^{(m)}$ with the following inside recursions.

$$
\begin{aligned}
I(\Gamma^{(1)}_{n,d}) &= I(S^{(1)}_{0,|x(1)|,n,d}) \\
I(\Gamma^{(\ell)}_{n,d}) &= \sum_{n',n'':n'+n''=n} I(\Gamma^{(\ell-1)}_{n',d'}) I(S^{(\ell)}_{0,N,n'',d''}) \\
I(GOAL) &= \sum_{n,d} I(\Gamma^{(m)}_{n,d}) \left(1 - \frac{2n}{d+|y|}\right)
\end{aligned}
$$

Outside recursions follow straightforwardly. Implementation of this algorithm would require substantial distributed computation or external data structures, so we did not attempt it.

References

M. Auli and A. Lopez. 2011. A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing. In *Proc. of ACL*, June.

J. K. Baker. 1979. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65.

S. Bangalore and A. K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):238–265, June.

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

S. Clark and J. R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *COLING*, Morristown, NJ, USA.

S. Clark and J. R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

S. Clark and J. Hockenmaier. 2002. Evaluating a Wide-Coverage CCG Parser. In *Proceedings of the LREC 2002 Beyond Parseval Workshop*, pages 60–66, Las Palmas, Spain.

S. Clark, J. Hockenmaier, and M. Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proc. of ACL*.

S. Clark. 2002. Supertagging for Combinatory Categorial Grammar. In *TAG+6*.

M. Dreyer and J. Eisner. 2009. Graphical models over multiple strings. In *Proc. of EMNLP*.

J. R. Finkel, A. Kleeman, and C. D. Manning. 2008. Feature-based, conditional random field parsing. In *Proceedings of ACL-HLT*.

T. A. D. Fowler and G. Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proc. of ACL*.

K. Gimpel and N. A. Smith. 2010a. Softmax-margin CRFs: training log-linear models with cost functions. In *HLT '10: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

K. Gimpel and N. A. Smith. 2010b. Softmax-margin training for structured log-linear models. Technical Report CMU-LTI-10-008, Carnegie Mellon University.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*, pages 177–183, Jun.

J. Hockenmaier and M. Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

L. Huang. 2008. Forest Reranking: Discriminative parsing with Non-Local Features. In *Proceedings of ACL-08: HLT*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289.

D. McAllester, M. Collins, and F. Pereira. 2008. Case-factor diagrams for structured probabilistic modeling. *Journal of Computer and System Sciences*, 74(1):84–96.

D. McAllester. 1999. On the complexity analysis of static analyses. In *Proc. of Static Analysis Symposium*, volume 1694/1999 of *LNCS*. Springer Verlag.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, Jul.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

D. Povey and P. Woodland. 2008. Minimum phone error and I-smoothing for improved discrimative training. In *Proc. of ICASSP*.

F. Sha and L. K. Saul. 2006. Large margin hidden Markov models for automatic speech recognition. In *Proc. of NIPS*.

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

M. Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA.

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

Large-Scale Cognate Recovery

David Hall and Dan Klein
Computer Science Division
University of California at Berkeley
{dlwh,klein}@cs.berkeley.edu

Abstract

We present a system for the large scale induction of cognate groups. Our model explains the evolution of cognates as a sequence of mutations and innovations along a phylogeny. On the task of identifying cognates from over 21,000 words in 218 different languages from the Oceanic language family, our model achieves a cluster purity score over 91%, while maintaining pairwise recall over 62%.

1 Introduction

The critical first step in the reconstruction of an ancient language is the recovery of related *cognate* words in its descendants. Unfortunately, this process has largely been a manual, linguistically-intensive undertaking for any sizable number of descendant languages. The traditional approach used by linguists—the comparative method—iterates between positing putative cognates and then identifying regular sound laws that explain correspondences between those words (Bloomfield, 1938).

Successful computational approaches have been developed for large-scale reconstruction of phylogenies (Ringe et al., 2002; Daumé III and Campbell, 2007; Daumé III, 2009; Nerbonne, 2010) and ancestral word forms of known cognate sets (Oakes, 2000; Bouchard-Côté et al., 2007; Bouchard-Côté et al., 2009), enabling linguists to explore deep historical relationships in an automated fashion. However, computational approaches thus far have not been able to offer the same kind of scale for identifying cognates. Previous work in cognate identification has largely focused on identifying cognates in pairs of languages (Mann and Yarowsky, 2001; Lowe and Mazaudon, 1994; Oakes, 2000; Kondrak, 2001; Mulloni, 2007), with a few recent exceptions that can find sets in a handful of languages (Bergsma and Kondrak, 2007; Hall and Klein, 2010).

While it may seem surprising that cognate detection has not successfully scaled to large numbers of languages, the task poses challenges not seen in reconstruction and phylogeny inference. For instance, morphological innovations and irregular sound changes can completely obscure relationships between words in different languages. However, in the case of reconstruction, an unexplainable word is simply that: one can still correctly reconstruct its ancestor using words from related languages.

In this paper, we present a system that uses two generative models for large-scale cognate identification. Both models describe the evolution of words along a phylogeny according to automatically learned sound laws in the form of parametric edit distances. The first is an adaptation of the generative model of Hall and Klein (2010), and the other is a new generative model called PARSIM with connections to parsimony methods in computational biology (Cavalli-Sforza and Edwards, 1965; Fitch, 1971). Our model supports simple, tractable inference via message passing, at the expense of being unable to model some cognacy relationships. To help correct this deficiency, we also describe an agglomerative inference procedure for the model of Hall and Klein (2010). By using the output of our system as input to this system, we can find cognate groups that PARSIM alone cannot recover.

We apply these models to identifying cognate groups from two language families using the Austronesian Basic Vocabulary Database (Greenhill et al., 2008), a catalog of words from about 40% of the Austronesian languages. We focus on data from two subfamilies of Austronesian: Formosan and

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 344–354,
Edinburgh, Scotland, UK, July 27–31, 2011.

Oceanic. The datasets are by far the largest on which automated cognate recovery has ever been attempted, with 18 and 271 languages respectively. On the larger Oceanic data, our model can achieve cluster purity scores of 91.8%, while maintaining pairwise recall of 62.1%. We also analyze the mistakes of our system, where we find that some of the erroneous cognate groups our system finds may not be errors at all. Instead, they may be previously unknown cognacy relationships that were not annotated in the data.

2 Background

Before we present our model, we first describe basic facts of the Austronesian language family, along with a description of the Austronesian Basic Vocabulary Database, which forms the dataset that we use for our experiments. For far more detailed coverage of the Austronesian languages, we direct the interested reader to Blust (2009)'s comprehensive monograph.

2.1 The Austronesian Language Family

The Austronesian language family is one of the largest in the world, comprising about one-fifth of the world's languages. Geographically, it stretches from its homeland on Formosa (Taiwan) to Madagascar in the west, and as far as Hawai'i and (at one point) the Easter Islands to the east. Until the advent of European colonialism spread Indo-European languages to every continent, Austronesian was the most widespread of all language families.

Linguistically, the language family is as diverse as it is large, but a few regularities hold. From a phonological perspective, two features stand out. First, the phoneme inventories of these languages are typically small. For example, it is well-known that Hawaiian has only 13 phonemes. Moreover, the phonotactics of these languages are often restrictive. Sticking with the same example, Hawaiian only allows (C)V syllables: consonants clusters are forbidden, and no syllable may end with a consonant.

2.2 The Austronesian Basic Vocabulary Database

The Austronesian Basic Vocabulary Database (ABVD) (Greenhill et al., 2008) is an ambitious, ongoing effort to catalog the lexicons and basic facts about all of the languages in the Austronesian language family. It also contains manual reconstructions for select ancestor languages produced by linguists.

The sample we use—from Bouchard-Côté et al. (2009)—contains about 50,000 words across 471 languages spanning all the major divisions of Austronesian. These words are grouped into cognate groups and arranged by gloss. For instance, there are 37 distinct cognate groups for the gloss "tail." One of these groups includes the words */ekor/*, */ingko/*, */iŋkot/*, */kiiki?u/*, and */?i?ina/*, among others. Most of these words have been transcribed into the International Phonetic Alphabet, though it appears that some words are transcribed using the Roman alphabet. For instance, the second word in the example is likely */iŋko/*, which is a much more likely sequence than what is transcribed.

In this sample, there are 6307 such cognate groups and 210 distinct glosses. The data is somewhat sparse: fewer than 50% of the possible gloss/language pairs are present. Moreover, there is some amount of *homoplasy*—that is, languages with a word from more than one cognate group for a given gloss.

Finally, it is important to note that the ABVD is still a work in progress: they have data from only 50% of extant Austronesian languages.

2.3 Subfamilies of Austronesian

In this paper we focus on two branches of the Austronesian language family, one as a development set and one as a test set. For our development set, we use the Formosan branch. The languages in this group are exclusively found on the Austronesian homeland of Formosa. The family encompasses a substantial portion of the linguistic diversity of Austronesian: Blust (2009) argues that Formosan contains 9 of the 10 first-order splits of the Austronesian family. Formosan's diversity is surprising since it contains a mere 18 languages. Thus, Formosan is a smaller development set that nevertheless is representative of larger families.

For our final test set, we use the Oceanic subfamily, which includes almost 50% of the languages in the Austronesian family, meaning that it represents around 10% of all languages in the world. Oceanic also represents a large fraction of the ge-

ographic diversity of Austronesian, stretching from New Zealand in the south to Hawai'i in the north. Our sample includes 21863 words from 218 languages in the Oceanic family.

3 Models

In this section we describe two models, one based on Hall and Klein (2010)—which we call HK10—and another new model that shares some connection to parsimony methods in computational biology, which we call PARSIM. Both are generative models that describe the evolution of words w_ℓ from a set of languages $\{\ell\}$ in a cognate group g along a fixed phylogeny T.[1] Each cognate group and word is also associated with a gloss or meaning m, which we assume to be fixed.[2] In both models, words evolve according to regular sound laws φ_ℓ, which are specific to each language. Also, both models will make use of a language model λ, which is used for generating words that are not dependent on the word in the parent language. (We leave φ_ℓ and λ as abstract parameters for now. We will describe them in subsequent sections.)

3.1 HK10

The first model we describe is a small modification of the phylogenetic model of Hall and Klein (2010). In HK10, there is an unknown number of cognate groups G where each cognate group g consists of a set of words $\{w_{g,\ell}\}$. In each cognate group, words evolve along a phylogeny, where each word in a language is the result of that word evolving from its parent according to regular sound laws. To model the fact that not all languages have a cognate in each group, each language in the tree has an associated "survival" variable $S_{g,\ell}$, where a word may be lost on that branch (and its descendants) instead of evolving. Once the words are generated, they are then "permuted" so that the cognacy relationships

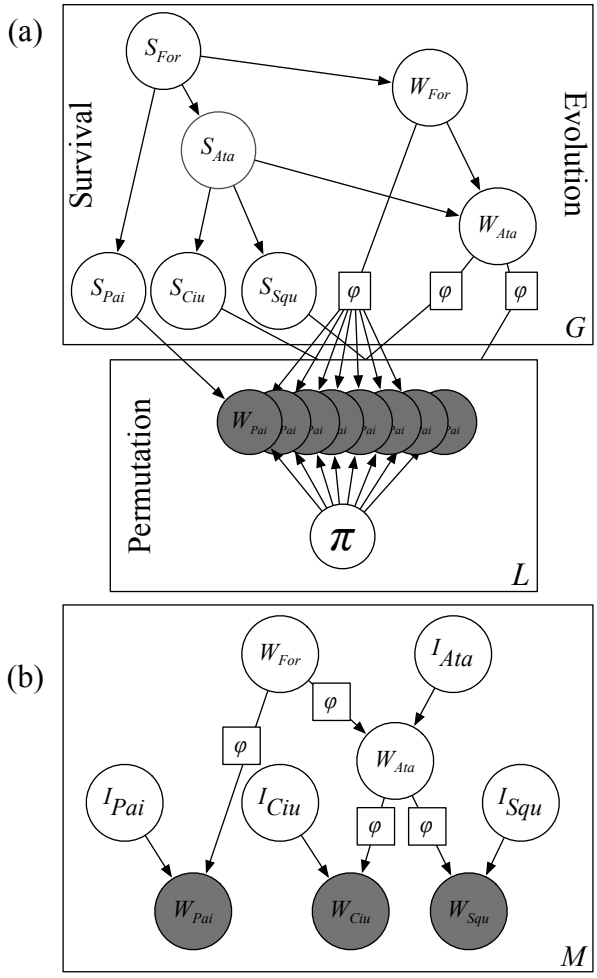


Figure 1: Plate diagrams for (a) HK10 (Hall and Klein, 2010) and (b) PARSIM, our new parsimony model, for a small set of languages. In HK10, words are generated following a phylogenetic tree according to sound laws φ, and then "scrambled" with a permutation π so that the original cognate groups are lost. In PARSIM, all words for each of the M glosses are generated in a single tree, with innovations I starting new cognate groups. The languages depicted are Formosan (For), Paiwan (Pai), Atayalic (Ata), Ciuli Atayalic (Ciu), and Squliq Atayalic (Squ).

[1] Both of these models therefore are insensitive to geographic and historical factors that cannot be easily approximated by this tree. See Nichols (1992) for an excellent discussion of these factors.

[2] One could easily envision allowing the meaning of a word to change as well. Modeling this semantic drift has been considered by Kondrak (2001). In the ABVD, however, any semantic drift has already been elided, since the database has coarsened glosses to the extent that there is no meaningful way to model semantic drift given our data.

are obscured. The task of inference then is to recover the original cognate groups.

The generative process for their model is as follows:

- For each cognate group g, choose a root word $W_{root} \sim p(W|\lambda)$, a language model over words.
- For each language ℓ in a pre-order traversal of the phylogeny:
  1. Choose $S_\ell \sim \text{Bernoulli}(\beta_\ell)$, indicating whether or not the word survives.
  2. If the word survives, choose $W_\ell \sim p(W|\varphi_\ell, W_{\text{par}(\ell)})$.
  3. Otherwise, stop generating words in that language and its descendants.
- For each language, choose a random permutation π of the observed data, and rearrange the cognates according to this permutation.

We reproduce the graphical model for HK10 for a small phylogeny in Figure 1a.

Inference in this model is intractable; to perform inference exactly, one has to reason over all partitions of the data into cognate groups. To address this problem, Hall and Klein (2010) propose an iterative bipartite matching scheme where one language is held out from the others, and then words are assigned to the remaining groups to maximize the probability of the attachment. That is, for some language ℓ and fixed assignments $\pi_{-\ell}$ for the other languages, they seek an assignment π_ℓ that maximizes:

$$\pi^* = \underset{\pi}{\text{argmax}} \sum_g \log p(w_{(\ell,\pi_\ell(g))}|\varphi, \pi, \mathbf{w}_{-\ell})$$

Unfortunately, while this approach was effective with only a few languages (they tested on three), this algorithm cannot scale to the eighteen languages in Formosan, let alone the hundreds of languages in Oceanic. Therefore, we make two simple modifications. First, we restrict the cognate assignments to stay within a gloss. Thus, there are many fewer potential matchings to consider. Second, we use an agglomerative inference procedure, which greedily merges cognate groups that result in the greatest gain in likelihood. That is, for all pairs of cognate groups g_a with words $\mathbf{w}_a$ and g_b with words $\mathbf{w}_b$, we compute the score:

$$\log p(\mathbf{w}_{a\cup b}|\varphi) - \log p(\mathbf{w}_a|\varphi) - \log p(\mathbf{w}_b|\varphi)$$

This score is the difference between the log probability of generating two cognate groups jointly and generating them separately. We then merge the two that generate the highest gain in likelihood. Like the iterative bipartite matching algorithm described above, this algorithm is not exact. However, it is $O(n^2 \log n)$ (where n is the size of the largest gloss, which for Oceanic is 153), while the bipartite matching algorithm is $O(n^3)$ (Kuhn, 1955).

Actually, the original HK10 is doubly intractable. They use weighted automata to represent distributions over strings, but these automata—particularly if they are non-deterministic—make inference in any non-trivial graphical model intractable. We discuss this issue in more detail in Section 6.

3.2 A Parsimony-Inspired Model

We now describe a new model called PARSIM that supports exact inference tractably, though it sacrifices some of the expressive power of HK10. In our model, each language has at most one word for each gloss, and this one word changes from one language to its children according to some edge-specific Markov process. These changes may either be mutations, which merely change the surface form of the word, or innovations, which start a new word in a new cognate group that is unrelated to the previous word. Mutations take the form of a conditional edit operation that models insertions, substitutions, and deletions that correspond to regular (and, with lower probability, irregular) sound changes that are likely to occur between a language and its parent. Innovations, on the other hand, are generated from a language model independent of the parent's word.

Specifically, our generative process takes the following form:

- For each gloss m, choose a root word $W_{root} \sim \lambda$, a language model over words.
- For each language ℓ in a pre-order traversal of the phylogeny:

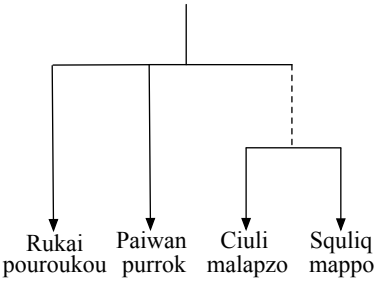


Figure 2: A small example of how PARSIM works. Listed here are the words for "ten" in four languages from the Formosan family, along with the tree that explains them. The dashed line indicates an innovation on the branch.

1. Choose $I_\ell \sim \text{Bernoulli}(\beta_\ell)$, indicating whether or not the word is an innovation or a mutation.
2. If it is a mutation, choose $W_\ell \sim p(W|\varphi_\ell, W_{\text{par}(\ell)})$.
3. Otherwise, choose $W_\ell \sim \lambda$.

We also depict our model as a plate diagram for a small phylogeny in Figure 1b.

Because there is only one tree per gloss, there is no assignment problem to consider, which is the main source of the intractability of HK10. Instead, pieces of the phylogeny are simply "cut" into subtrees whenever an innovation occurs. Thus, message passing can be used to perform inference.

As an example of how our process works, consider Figure 2. The Formosan word for "ten" probably resembled either */purrok/* or */pouroukou/*. There was an innovation in Ciuli and Squliq's ancestor Atayalic that produced a new word for ten. This word then mutated separately into the words */malapzo/* and */mappo/*, respectively.

4 Relation to Parsimony

PARSIM is related to the parsimony principle from computational biology (Cavalli-Sforza and Edwards, 1965; Fitch, 1971), where it is used to search for phylogenies. When using parsimony, a phylogeny is scored according to the derivation that requires the fewest number of changes of state, where a state is typically thought of as a gene or some other trait in a species. These genes are typically called "characters" in the computational biology literature, and two species would have the same value for a character if they share the same property that that state represents.

When inducing phylogenies of languages, a natural choice for characters are glosses from a restricted vocabulary like a Swadesh list, and two words are represented as the same value for a character if they are cognate (Ringe et al., 2002). Other features can be used (Daumé III and Campbell, 2007; Daumé III, 2009), but they are not relevant to our discussion.

Consider the small example in Figure 3a with just four languages. Here, cognacy is encoded using characters. In this example, at least two changes of state are required to explain the data: both C and B must have evolved from A. Therefore, the parsimony score for this tree is two.

Of course, there is no reason why all changes should be equally likely. For instance, it might be extremely likely that B changes into both A and C, but that A never changes into B or C, and so weighted variants of parsimony might be necessary (Sankoff and Cedergren, 1983).

With this in mind, PARSIM can be thought of a weighted variant of parsimony, with two differences. First, the characters do not indicate ahead of time which words are related. Instead, the characters are the words themselves. Second, the transitions between different states (words) are not uniform. Instead, they are weighted by the log probability of one word changing into another, including both mutations and innovations.

Thus, the task of inference in PARSIM is to find the most "parsimonious" explanation for the words we have observed, which is the same as finding the most likely derivation. Because the distances between words (that is, the transition probabilities) are not known ahead of time, they must instead be learned, which we discuss in Section 7.[3]

5 Limitations of the Parsimony Model

Potentially, our parsimony model sacrifices a certain amount of power to make inference tractable. Specifically, it cannot model *homoplasy*, the presence of more than one word in a language for a given

[3]It is worth noting that we are not the first to point out a connection between parsimony and likelihood. Indeed, many authors in the computational biology literature have formally demonstrated a connection (Farris, 1973; Felsenstein, 1973).

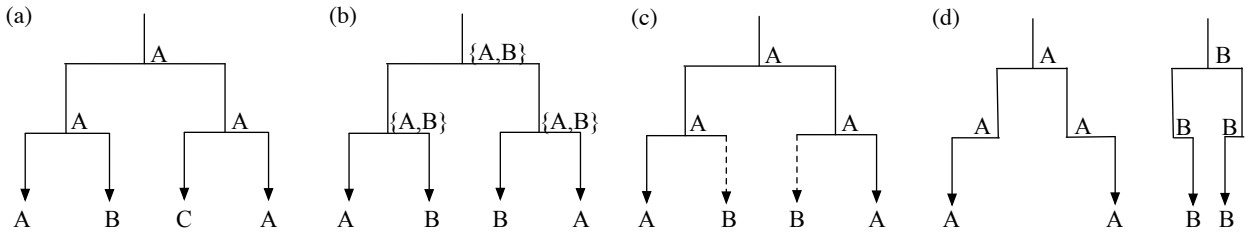


Figure 3: Trees illustrating parsimony and its limitations. In these trees, there are four languages, with words A, B, and C in various configurations. (a) The most parsimonious derivation for this tree has all intermediate states as A. There are thus two changes. (b) An example of homoplasy. Here, given this tree, it seems likely that the ancestral languages contained both A and B. (c) PARSIM cannot recover the example from (b), and so it encodes two innovations (shown as dashed lines). (d) The HK10 model can recover this relationship, but this power makes the model intractable.

gloss. Homoplasy can arise for a variety of reasons in phylogenetic models of cognates, and we describe some in this section.

Consider the example illustrated in Figure 3b, where the two central languages share a cognate, as do the two outer languages. This is the canonical example of homoplasy, and PARSIM cannot correctly recover this grouping. Instead, it can at best only select group A or group B as the value for the parent, and leave the other group fragmented as two innovations, as in Figure 3c. On the other hand, HK10 *can* recover this relationship (Figure 3d), but this power is precisely what makes it intractable.

There are two reasons this kind of homoplasy could arise. The first is that there were indeed two words in the parent language for this gloss, or that there were two words with similar meanings and the two meanings drifted together. Second, the tree could be an inadequate model of the evolution in this case. For instance, there could have been a certain amount of borrowing between two of these languages, or there was not a single coherent parent language, but rather a language continuum that cannot be explained by any tree.

However, homoplasy seems to be relatively uncommon (though not unheard of) in the Oceanic and Formosan families. Where it does appear, our model should simply fail to get one of the cognate groups, instead explaining all of them via innovation. To repair this shortcoming, we can simply run the agglomerative clustering procedure for the model of Hall and Klein (2010), starting from the groups that PARSIM has recovered. Using this procedure, we can hopefully recover many of the under-groupings caused by homoplasy.

6 Inference and Scale

6.1 Inference

In this section we describe the basics of inference in the PARSIM model. We have a nearly tree-structured graphical model (Figure 1); it is not a tree only because of the innovation parameters. Therefore, we apply the common trick of grouping variables to form a tree. Specifically, we group each word variable W_ℓ with its innovation parameter I_ℓ. The distribution of interest is then $p(W_\ell, I_\ell | W_{\mathbf{par}(\ell)}, \phi_\ell, \beta_\ell)$, and the primary operation is summing out messages μ from the children of a language and sending a new message to its parent:

$$\mu_\ell(w_{\mathbf{par}(\ell)}) = \sum_{w_\ell} p(w_\ell|\cdot) \prod_{\ell' \in \mathbf{child}(\ell)} \mu_{\ell'}(w_\ell)$$
$$p(w_\ell|\cdot) = p(w_\ell|I_\ell = 0, w_{\mathbf{par}(\ell)}, \phi_\ell) p(I_\ell = 0|\beta_\ell) + p(w_\ell|I_\ell = 1, \phi_\ell) P(I_\ell = 1|\beta_\ell) \quad (1)$$

The first term involves computing the probability of the word mutating from its parent, and the second involves the probability of the child word from a language model. We describe the parameters and procedures for these operations in 7.1.

6.2 Scale

Even though inference by message-passing in our model is tractable, we needed to make certain concessions to make inference acceptably fast. These choices mainly affect how we represent distributions over strings.

First, we need to model distributions and messages over words on the internal nodes of a phylogeny. The natural choice in this scenario is to use weighted finite automata (Mohri et al., 1996). Automata have been used to successfully model distributions of strings for inferring morphology (Dreyer and Eisner, 2009) as well as cognate detection (Hall and Klein, 2010). Even in models that would be tractable with "ordinary" messages, inference with automata quickly becomes intractable, because the size of the automata grow exponentially with the number of messages passed. Therefore, approximations must be used. Dreyer and Eisner (2009) used a mixture of a k-best list and a unigram language model, while Hall and Klein (2010) used an approximation procedure that projected complex automata to simple, tractable automata using a modified KL divergence.

While either approach could be used here in principle, we found that automata machinery was simply too slow for our application. Instead, we exploit the intuition that we do not need to accurately reconstruct the word for any ancestral language. Moreover, it is inefficient to keep track of probabilities for all strings. Therefore, we only track scores for words that actually exist in a given gloss, which means that internal nodes only have mass on those words. That is, if a gloss has 10 distinct words across all the languages in our dataset, we pass messages that only contain information about those 10 words.

Now, this representation—while more efficient than the automata representations—results in inference that is still quadratic in the number of words in a gloss, since we have distributions of the form $p(w_\ell | w_{\mathrm{par}(\ell)}, \phi_\ell)$. Intuitively, it is unlikely that a word from one distant branch of tree resembles a word in another branch. Therefore, rather than score all of these unlikely words, we use a beam where we only factor in words whose score is at most a factor of e^{-10} less than the maximum score. Our initial experiments found that using a beam provides large savings in time with little impact on prediction quality.

7 Learning

PARSIM has three kinds of parameters that we need to learn: the mutation parameters φ_ℓ, the innovation probabilities β_ℓ, and the global language model λ for generating new words. We learn these parameters via Expectation Maximization (Dempster et al., 1977), iterating between computing expected counts and adjusting parameters to maximize the posterior probability of the parameters. In this section, we describe those parameters.

7.1 Sound Laws

The core piece of our system is learning the sound laws associated with each edge. Since the foundation of historical linguists with the neogrammarians, linguists have argued for the regularity of sound change at the phonemic level (Schleicher, 1861; Bloomfield, 1938). That is to say, if in some language a $/t/$ changes to a $/d/$ in some word, it is almost certain that it will change in every other place that has the same surrounding context.

In practice, of course, sound change is not entirely regular, and complex extralinguistic events can lead to sound changes that are irregular. For example, in some cultures in which Oceanic languages are spoken, the name of the chief is taboo: one cannot speak his name, nor say any word that sounds too much like his name. Speakers of these languages do find ways around this prohibition, often resulting in sound changes that cannot be explained by sound laws alone (Keesing and Fifi'i, 1969).

Nevertheless, we find it useful to model sound change as a largely regular if stochastic process. We employ a sound change model whose expressive power is equivalent to that of Hall and Klein (2010), though with a different parameterization. We model the evolution of a word w_ℓ to its child $w_{\ell'}$ as a sequence of unigram edits that include insertions, deletions, and substitutions. Specifically, we use a standard three-state pair hidden Markov model that is closely related to the classic alignment algorithm of Needleman and Wunsch (1970) (Durbin et al., 2006).

The three states in this HMM correspond to matches/substitutions, insertions, and deletions. The transitions are set up such that insertions and deletions cannot be interleaved. This prevents spurious equivalent alignments, which would cause the model to assign unnecessarily higher probability to transitions with many insertions and deletions.

Actually learning these parameters involves learn-

ing the transition probabilities of this HMM (which model the overall probability of insertion and deletion) as well as the emission probabilities (which model the particular edits). Because there are relatively few words for each language (96 on average in Oceanic), we found it important to tie together the parameters for the various languages, in contrast to Hall and Klein (2010) who did not. In our maximization step, we fit a joint log-linear model for each language, using features that are both specific to a language and shared across languages. Our features included indicators on each substitution, insertion, and deletion operation, along with an indicator for the outcome of each edit operation. This last feature reflects the propensity of a particular phoneme to appear in a given language at all, no matter what its ancestral phoneme was. This parameterization is similar to the one used in the reconstruction system of Bouchard-Côté et al. (2009), except that they used edit operations that conditioned on the context of the surrounding word, which is crucial when trying to accurately reconstruct ancestral word forms. To encourage parameter sharing, we used an ℓ_2 regularization penalty.

7.2 Innovation Parameters

The innovation parameters β_ℓ are parameters for simple Bernoulli distribution that govern the propensity for a language to start a new word. These parameters can be learned separately, though due to data sparsity, we found it better to use a tied parameterization as with the sound laws. Specifically, we fit a log linear model whose features are indicators on the specific language, as well as a global innovation parameter that is shared across all languages. As with the sound laws, we used an ℓ_2 regularization penalty to encourage the use of the global innovation parameter.

7.3 Language Model

Finally, we have a single language model λ that is also shared across all languages. λ is a simple bigram language model over characters in the International Phonetic Alphabet. λ is used when generating new words either via innovation or from the root of the tree.

In principle, we could of course have language models specific to each language, but because there are so few words per language, we found that branch-specific language models caused the model to prefer to innovate at almost every node since the language models could essentially memorize the relatively small vocabularies of these languages.

Formosan				
System	Prec	Recall	F1	Purity
Agg. HK10	77.6	**83.2**	80.0	84.7
PARSIM	**87.8**	71.0	78.5	**94.6**
Combination	85.2	81.3	**83.2**	92.3
Oceanic				
System	Prec	Recall	F1	Purity
PARSIM	**84.4**	62.1	71.5	**91.8**
Combination	76.0	73.8	**74.9**	85.5

Table 1: Results on the Formosan and Oceanic families. PARSIM is the new parsimony model in this paper, Agg. HK10 is our agglomerative variant of Hall and Klein (2010) and Combination uses PARSIM's output to seed the agglomerative matcher. For the agglomerative systems, we report the point with maximal F1 score, but we also show precision/recall curves. (See Figure 4.)

8 Experiments

8.1 Cognate Recovery

We ran both PARSIM and our agglomerative version of HK10 on the Formosan datasets. For PARSIM, we initialized the mutation parameters φ to a model that preferred matches to insertions, substitutions and deletions by a factor of e^3, innovation parameters to 0.5, and the language model to a uniform distribution over characters. For the agglomerative HK10, we initialized its parameters to the values found by our model.[4]

Based on our observations about homoplasy, we also considered a combined system where we ran PARSIM, and then seeded the agglomerative clustering algorithm with the clusters found by PARSIM.

For evaluation, we report a few metrics. First, we report cluster purity, which is a kind of precision measure for clusterings. Specifically, each cluster is assigned to the cognate group that is the most common cognate word in that group, and then purity is computed as the fraction of words that

[4] Attempts to learn parameters directly with the agglomerative clustering algorithm were not effective.

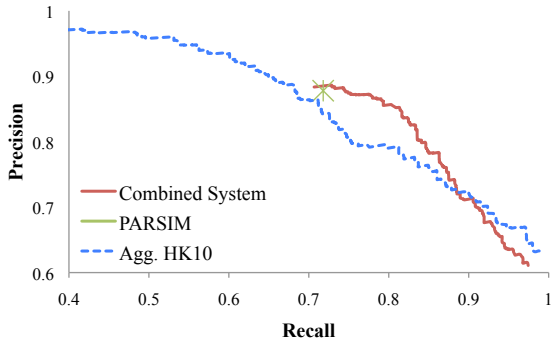


Figure 4: Precision/Recall curves for our systems. The Combined System starts from PARSIM's output, so it has fewer points to plot, and starts from a point with lower precision. As PARSIM outputs only one result, it is starred.

are in a cluster whose gold cognate group matches the cognate group of the cluster. For gold partitions $G = \{G_1, G_2, \ldots, G_g\}$ and found partitions $F = \{F_1, F_2, \ldots, F_f\}$, we have: $\text{purity}(G, F) = \frac{1}{N} \sum_f \max_g |G_g \cap F_f|$. We also report pairwise precision and recall computed over pairs of words.[5] Finally, because agglomerative clustering does not define a natural "stopping point" other than when the likelihood gain decreases to 0—which did not perform well in our initial tests—we will report both a precision/recall curve, as well the maximum pairwise F1 obtained by the agglomerative HK10 and the combined system.

The results are in Table 1. On Formosan, PARSIM has much higher precision and purity than our agglomerative version of HK10 at its highest point, though its recall and F1 suffer somewhat. Of course, the comparison is not quite fair, since we have selected the best possible point for HK10.

However, our combination of the two systems does even better. By feeding our high-precision results into the agglomerative system and sacrificing just a little precision, our combined system achieves much higher F1 scores than either of the systems alone.

Next, we also examined precision and recall curves for the two agglomerative systems on Formosan, which we have plotted in Figure 4, along with the one point output by PARSIM.

We then ran PARSIM and the combined system on the much larger Oceanic dataset. Performance on all metrics decreased somewhat, but this is to be expected since there is so much more data. As with Formosan, PARSIM has higher precision than the combined system, but it has much lower recall.

8.2 Reconstruction

We also wanted to see how well our cognates could be used to actually reconstruct the ancestral forms of words. To do so, we ran a version of Bouchard-Côté et al. (2009)'s reconstruction system using both the cognate groups PARSIM found in the Oceanic language family and the gold cognate groups provided by the ABVD. We then evaluated the average Levenshtein distance of the reconstruction for each word to the reconstruction of that word's Proto-Oceanic ancestor provided by linguists. Our evaluation differs from Bouchard-Côté et al. (2009) in that they averaged over cognate groups, which does not make sense for our task because there are different cognate groups. Instead, we average over per-modern-word reconstruction error.

Using this metric, reconstructions using our system's cognates are an average of 2.47 edit operations from the gold reconstruction, while with gold cognates the error is 2.19 on average. This represents an error increase of 12.8%. To see if there was some pattern to these errors, we also plotted the fraction of words with each Levenshtein distance for these reconstructions in Figure 5. While the plots are similar, the automatic cognates exhibit a longer tail. Thus, even with automatic cognates, the reconstruction system can reconstruct words faithfully in many cases, but in a few instances our system fails.

9 Analysis

We now consider some of the errors made by our system. Broadly, there are two kinds of mistakes in a model like ours: those affecting precision and those affecting recall.

9.1 Precision

Many of our precision errors seem to be due to our somewhat limited model of sound change. For instance, the language Pazeh has two words for

[5]The main difference between precision and purity is that pairwise precision is inherently quadratic, meaning that it penalizes mistakes in large groups much more heavily than mistakes in small groups.

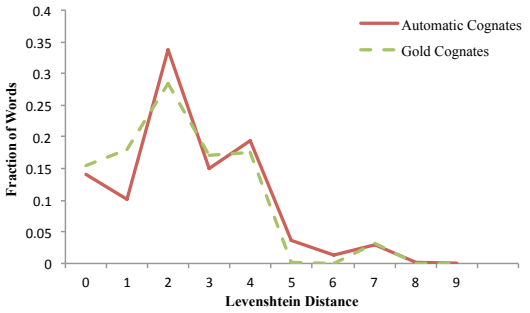


Figure 5: Percentage of words with varying levels of Levenshtein distance from the gold reconstruction. Gold Cognates were hand-annotated by linguists, while Automatic Cognates were found by our system.

"to sleep:" */mudamai/* and */midəm/*. Somewhat surprisingly the former word is cognate with Paiwan */qmereŋ/* and Saisiat */maʔrəm/* while the latter is not. Our system, however, makes the mistake of grouping */midəm/* with the Paiwan and Saisiat words. Our system has inferred that the insertions of */u/* and */ai/* (which are required to bring */mudamai/* into alignment with the Saisiat and Paiwan words) are less likely than substituting a few vowels and the consonant */r/* for */d/* (which are required to align */midəm/*). Perhaps a more sophisticated model of sound change could correctly learn this relationship.

However, a preliminary inspection of the data seems to indicate that not all of our precision errors are actually errors, but rather places where the data is insufficiently annotated (and indeed, the ABVD is still a work in progress). For instance, consider the words for "meat/flesh" in the Formosan languages: Squliq */hiʔ/*, Bunun */titiʔ/*, Paiwan */seti/*, Kavalan */ʔisiʔ/*, CentralAmi */titi/*, Our system groups all of these words except for Squliq */hiʔ/*. However, despite these words' similarity, there are actually three cognate groups here. One includes Squliq */hiʔ/* and Kavalan */ʔisiʔ/*, another includes just Paiwan */seti/*, and the third includes Bunun */titiʔ/* and CentralAmi */titi/*. Crucially, these cognate groups do not follow the phylogeny closely. Thus, either there was a significant amount of borrowing between these languages, or there was a striking amount of homoplasy in Proto-Formosan, or these words are in fact mostly cognate. While a more thorough, linguistically-informed analysis is needed to ensure that these are actually cognates, we believe that our system, in conjunction with a trained Austronesian specialist, could potentially find many more cognate groups, speeding up the process of completing the ABVD.

9.2 Recall

Our system can also fail to group words that should be grouped. One recurring problem seems to be reduplication, which is a fairly common phenomenon in Austronesian languages. For instance, there is a cognate group for "to eat" that includes Bunun */maun/*, Thao */kman/*, Favorlang */man/*, and Sediq */manakamakan/*, among others. Our system correctly finds this group, with the exception of */manakamakan/*, which is clearly the result of reduplication. Reduplication cannot be modeled using mere sound laws, and so a more complex transition model is needed to correctly identify these kinds of changes.

10 Conclusion

We have presented a new system for automatically finding cognates across many languages. Our system is comprised of two parts. The first, PARSIM, is a new high-precision generative model with tractable inference. The second, HK10, is a modification of Hall and Klein (2010) that makes their approximate inference more efficient. We discuss certain trade-offs needed to make both models scale, and demonstrated its performance on the Formosan and Oceanic language families.

References

Shane Bergsma and Greg Kondrak. 2007. Multilingual cognate identification using integer linear programming. In *RANLP Workshop on Acquisition and Management of Multilingual Lexicons*, Borovets, Bulgaria, September.

Leonard Bloomfield. 1938. *Language*. Holt, New York.

R. A. Blust. 2009. *The Austronesian languages*. Australian National University.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *EMNLP*.

Alexandre Bouchard-Côté, Thomas L. Griffiths, and Dan Klein. 2009. Improved reconstruction of protolanguage word forms. In *NAACL*, pages 65–73.

L. L. Cavalli-Sforza and A. W. F. Edwards. 1965. Analysis of human evolution. In S. J. Geerts Genetics Today,

editor, *Proceedings of XIth International Congress of Genetics, 1963, Vol*, page 923–933. 3, 3.

Hal Daumé III and Lyle Campbell. 2007. A Bayesian model for discovering typological implications. In *Conference of the Association for Computational Linguistics (ACL)*.

Hal Daumé III. 2009. Non-parametric Bayesian areal linguistics. In *NAACL*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *EMNLP*, Singapore, August.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. 2006. *Biological sequence analysis*. eleventh edition.

James S. Farris. 1973. On Comparing the Shapes of Taxonomic Trees. *Systematic Zoology*, 22(1):50–54, March.

J. Felsenstein. 1973. Maximum likelihood and minimum steps methods for estimating evolutionnary trees from data on discrete characters. *Systematic Zoology*, 23:240–249.

W. M. Fitch. 1971. Toward defining the course of evolution: minimal change for a specific tree topology. *Systematic Zoology*, 20:406–416.

S.J. Greenhill, R. Blust, and R.D. Gray. 2008. The Austronesian basic vocabulary database: from bioinformatics to lexomics. *Evolutionary Bioinformatics*, 4:271–283.

David Hall and Dan Klein. 2010. Finding cognates using phylogenies. In *Association for Computational Linguistics (ACL)*.

Robert M. Keesing and Jonathan Fifi'i. 1969. Kwaio word tabooing in its cultural context. *Journal of the Polynesian Society*, 78(2):154–177.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *NAACL*.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

John B. Lowe and Martine Mazaudon. 1994. The reconstruction engine: a computer implementation of the comparative method. *Computational Linguistics*, 20(3):381–417.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *NAACL*.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In *ECAI-96 Workshop*. John Wiley and Sons.

Andrea Mulloni. 2007. Automatic prediction of cognate orthography using support vector machines. In *ACL*, pages 25–30.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453.

John Nerbonne. 2010. Measuring the diffusion of linguistic change. *Philosophical Transactions of the Royal Society B: Biological Sciences*.

J. Nichols. 1992. *Linguistic diversity in space and time*. University of Chicago Press.

Michael P. Oakes. 2000. Computer estimation of vocabulary in a protolanguage from word lists in four daughter languages. *Quantitative Linguistics*, 7(3):233–243.

Don Ringe, Tandy Warnow, and Ann Taylor. 2002. Indo-european and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129.

D. Sankoff and R. J. Cedergren, 1983. *Simultaneuous comparison of three or more sequences related by a tree*, page 253–263. Addison-Wesley, Reading, MA.

August Schleicher. 1861. *A Compendium of the Comparative Grammar of the Indo-European, Sanskrit, Greek and Latin Languages.*

Domain Adaptation via Pseudo In-Domain Data Selection

Amittai Axelrod
University of Washington
Seattle, WA 98105
amittai@uw.edu

Xiaodong He
Microsoft Research
Redmond, WA 98052
xiaohe@microsoft.com

Jianfeng Gao
Microsoft Research
Redmond, WA 98052
jfgao@microsoft.com

Abstract

We explore efficient domain adaptation for the task of statistical machine translation based on extracting sentences from a large general-domain parallel corpus that are most relevant to the target domain. These sentences may be selected with simple cross-entropy based methods, of which we present three. As these sentences are not themselves identical to the in-domain data, we call them *pseudo in-domain* subcorpora. These subcorpora – 1% the size of the original – can then used to train small domain-adapted Statistical Machine Translation (SMT) systems which outperform systems trained on the entire corpus. Performance is further improved when we use these domain-adapted models in combination with a true in-domain model. The results show that more training data is not always better, and that best results are attained via proper domain-relevant data selection, as well as combining in- and general-domain systems during decoding.

1 Introduction

Statistical Machine Translation (SMT) system performance is dependent on the quantity and quality of available training data. The conventional wisdom is that more data is better; the larger the training corpus, the more accurate the model can be.

The trouble is that – except for the few all-purpose SMT systems – there is never enough training data that is directly relevant to the translation task at hand. Even if there is no formal genre for the text to be translated, any coherent translation task will have its own argot, vocabulary or stylistic preferences, such that the corpus characteristics will necessarily deviate from any all-encompassing model of language. For this reason, one would prefer to use more in-domain data for training. This would empirically provide more accurate lexical probabilities, and thus better target the task at hand. However, parallel in-domain data is usually hard to find[1], and so performance is assumed to be limited by the quantity of domain-specific training data used to build the model. Additional parallel data can be readily acquired, but at the cost of specificity: either the data is entirely unrelated to the task at hand, or the data is from a broad enough pool of topics and styles, such as the web, that any use this corpus may provide is due to its size, and not its relevance.

The task of domain adaptation is to translate a text in a particular (target) domain for which only a small amount of training data is available, using an MT system trained on a larger set of data that is not restricted to the target domain. We call this larger set of data a *general-domain* corpus, in lieu of the standard yet slightly misleading *out-of-domain* corpus, to allow a large uncurated corpus to include some text that may be relevant to the target domain.

Many existing domain adaptation methods fall into two broad categories. Adaptation can be done at the corpus level, by selecting, joining, or weighting the datasets upon which the models (and by extension, systems) are trained. It can be also achieved at the model level by combining multiple translation or language models together, often in a weighted manner. We explore both categories in this work.

[1] Unless one dreams of translating parliamentary speeches.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 355–362,
Edinburgh, Scotland, UK, July 27–31, 2011.

First, we present three methods for ranking the sentences in a general-domain corpus with respect to an in-domain corpus. A cutoff can then be applied to produce a very small–yet useful– subcorpus, which in turn can be used to train a domain-adapted MT system. The first two data selection methods are applications of language-modeling techniques to MT (one for the first time). The third method is novel and explicitly takes into account the bilingual nature of the MT training corpus. We show that it is possible to use our data selection methods to subselect less than 1% (or discard 99%) of a large general training corpus and still increase translation performance by nearly 2 BLEU points.

We then explore how best to use these selected subcorpora. We test their combination with the in-domain set, followed by examining the subcorpora to see whether they are actually in-domain, out-of-domain, or something in between. Based on this, we compare translation model combination methods.

Finally, we show that these tiny translation models for model combination can improve system performance even further over the current standard way of producing a domain-adapted MT system. The resulting process is lightweight, simple, and effective.

2 Related Work

2.1 Training Data Selection

An underlying assumption in domain adaptation is that a general-domain corpus, if sufficiently broad, likely includes some sentences that could fall within the target domain and thus should be used for training. Equally, the general-domain corpus likely includes sentences that are so unlike the domain of the task that using them to train the model is probably more harmful than beneficial. One mechanism for domain adaptation is thus to select only a portion of the general-domain corpus, and use only that subset to train a complete system.

The simplest instance of this problem can be found in the realm of language modeling, using perplexity-based selection methods. The sentences in the general-domain corpus are scored by their perplexity score according to an in-domain language model, and then sorted, with only the lowest ones being retained. This has been done for language modeling, including by Gao et al (2002), and more recently by Moore and Lewis (2010). The ranking of the sentences in a general-domain corpus according to in-domain perplexity has also been applied to machine translation by both Yasuda et al (2008), and Foster et al (2010). We test this approach, with the difference that we simply use the source side perplexity rather than computing the geometric mean of the perplexities over both sides of the corpus. We also reduce the size of the training corpus far more aggressively than Yasuda et al's 50%. Foster et al (2010) do not mention what percentage of the corpus they select for their *IR-baseline*, but they concatenate the data to their in-domain corpus and report a decrease in performance. We both keep the models separate and reduce their size.

A more general method is that of (Matsoukas et al., 2009), who assign a (possibly-zero) weight to each sentence in the large corpus and modify the empirical phrase counts accordingly. Foster et al (2010) further perform this on extracted phrase pairs, not just sentences. While this soft decision is more flexible than the binary decision that comes from including or discarding a sentence from the subcorpus, it does not reduce the size of the model and comes at the cost of computational complexity as well as the possibility of overfitting. Additionally, the most effective features of (Matsoukas et al., 2009) were found to be meta-information about the source documents, which may not be available.

Another perplexity-based approach is that taken by Moore and Lewis (2010), where they use the cross-entropy difference as a ranking function rather than just cross-entropy. We apply this criterion for the first time to the task of selecting training data for machine translation systems. We furthermore extend this idea for MT-specific purposes.

2.2 Translation Model Combination

In addition to improving the performance of a single general model with respect to a target domain, there is significant interest in using two translation models, one trained on a larger general-domain corpus and the other on a smaller in-domain corpus, to translate in-domain text. After all, if one has access to an in-domain corpus with which to select data from a general-domain corpus, then one might as well use the in-domain data, too. The expectation is that the larger general-domain model should dom-

inate in regions where the smaller in-domain model lacks coverage due to sparse (or non-existent) ngram counts. In practice, most practical systems also perform target-side language model adaptation (Eck et al., 2004); we eschew this in order to isolate the effects of translation model adaptation alone.

Directly concatenating the phrase tables into one larger one isn't strongly motivated; identical phrase pairs within the resulting table can lead to unpredictable behavior during decoding. Nakov (2008) handled identical phrase pairs by prioritizing the source tables, however in our experience identical entries in phrase tables are not very common when comparing across domains. Foster and Kuhn (2007) interpolated the in- and general-domain phrase tables together, assigning either linear or log-linear weights to the entries in the tables before combining overlapping entries; this is now standard practice.

Lastly, Koehn and Schroeder (2007) reported improvements from using multiple decoding paths (Birch et al., 2007) to pass both tables to the Moses SMT decoder (Koehn et al., 2003), instead of directly combining the phrase tables to perform domain adaptation. In this work, we directly compare the approaches of (Foster and Kuhn, 2007) and (Koehn and Schroeder, 2007) on the systems generated from the methods mentioned in Section 2.1.

3 Experimental Framework

3.1 Corpora

We conducted our experiments on the International Workshop on Spoken Language Translation (IWSLT) Chinese-to-English DIALOG task [2], consisting of transcriptions of conversational speech in a travel setting. Two corpora are needed for the adaptation task. Our in-domain data consisted of the IWSLT corpus of approximately 30,000 sentences in Chinese and English. Our general-domain corpus was 12 million parallel sentences comprising a variety of publicly available datasets, web data, and private translation texts. Both the in- and general-domain corpora were identically segmented (in Chinese) and tokenized (in English), but otherwise unprocessed. We evaluated our work on the 2008 IWSLT spontaneous speech Challenge Task[3] test set, consisting of 504 Chinese sentences with 7 English reference translations each. This is the most recent IWSLT test set for which the reference translations are available.

3.2 System Description

In order to highlight the data selection work, we used an out-of-the-box Moses framework using GIZA++ (Och and Ney, 2003) and MERT (Och, 2003) to train and tune the machine translation systems. The only exception was the phrase table for the large out-of-domain system trained on 12m sentence pairs, which we trained on a cluster using a word-dependent HMM-based alignment (He, 2007). We used the Moses decoder to produce all the system outputs, and scored them with the NIST `mt-eval31a` [4] tool used in the IWSLT evalutation.

3.3 Language Models

Our work depends on the use of language models to rank sentences in the training corpus, in addition to their normal use during machine translation tuning and decoding. We used the SRI Language Modeling Toolkit (Stolcke, 2002) was used for LM training in all cases: corpus selection, MT tuning, and decoding. We constructed 4gram language models with interpolated modified Kneser-Ney discounting (Chen and Goodman, 1998), and set the Good-Turing threshold to 1 for trigrams.

3.4 Baseline System

The in-domain baseline consisted of a translation system trained using Moses, as described above, on the IWSLT corpus. The resulting model had a phrase table with 515k entries. The general-domain baseline was substantially larger, having been trained on 12 million sentence pairs, and had a phrase table containing 1.5 billion entries. The BLEU scores of the baseline single-corpus systems are in Table 1.

Corpus	Phrases	Dev	Test
IWSLT	515k	45.43	37.17
General	1,478m	42.62	40.51

Table 1: Baseline translation results for in-domain and general-domain systems.

[2] `http://iwslt2010.fbk.eu/node/33`

[3] Correct-Recognition Result (CRR) condition

[4] `http://www.itl.nist.gov/iad/mig/tools/`

4 Training Data Selection Methods

We present three techniques for ranking and selecting subsets of a general-domain corpus, with an eye towards improving overall translation performance.

4.1 Data Selection using Cross-Entropy

As mentioned in Section 2.1, one established method is to rank the sentences in the general-domain corpus by their perplexity score according to a language model trained on the small in-domain corpus. This reduces the perplexity of the general-domain corpus, with the expectation that only sentences similar to the in-domain corpus will remain. We apply the method to machine translation, even though perplexity reduction has been shown to not correlate with translation performance (Axelrod, 2006). For this work we follow the procedure of Moore and Lewis (2010), which applies the cosmetic change of using the cross-entropy rather than perplexity.

The perplexity of some string s with empirical n-gram distribution p given a language model q is:

$$2^{-\sum_x p(x)\log q(x)} = 2^{H(p,q)} \quad (1)$$

where $H(p,q)$ is the *cross-entropy* between p and q. We simplify this notation to just $H_I(s)$, meaning the cross-entropy of string s according to a language model LM_I which has distribution q. Selecting the sentences with the lowest perplexity is therefore equivalent to choosing the sentences with the lowest cross-entropy according to the in-domain language model. For this experiment, we used a language model trained (using the parameters in Section 3.3) on the Chinese side of the IWSLT corpus.

4.2 Data Selection using Cross-Entropy Difference

Moore and Lewis (2010) also start with a language model LM_I over the in-domain corpus, but then further construct a language model LM_O of similar size over the general-domain corpus. They then rank the general-domain corpus sentences using:

$$H_I(s) - H_O(s) \quad (2)$$

and again taking the lowest-scoring sentences. This criterion biases towards sentences that are both *like* the in-domain corpus and *unlike* the average of the general-domain corpus. For this experiment we reused the in-domain LM from the previous method, and trained a second LM on a random subset of 35k sentences from the Chinese side of the general corpus, except using the same vocabulary as the in-domain LM.

4.3 Data Selection using Bilingual Cross-Entropy Difference

In addition to using these two monolingual criteria for MT data selection, we propose a new method that takes in to account the bilingual nature of the problem. To this end, we sum cross-entropy difference over each side of the corpus, both source and target:

$$[H_{I-src}(s) - H_{O-src}(s)] + [H_{I-tgt}(s) - H_{O-tgt}(s)] \quad (3)$$

Again, lower scores are presumed to be better. This approach reuses the source-side language models from Section 4.2, but requires similarly-trained ones over the English side. Again, the vocabulary of the language model trained on a subset of the general-domain corpus was restricted to only cover those tokens found in the in-domain corpus, following Moore and Lewis (2010).

5 Results of Training Data Selection

The baseline results show that a translation system trained on the general-domain corpus outperforms a system trained on the in-domain corpus by over 3 BLEU points. However, this can be improved further. We used the three methods from Section 4 to identify the best-scoring sentences in the general-domain corpus.

We consider three methods for extracting domain-targeted parallel data from a general corpus: source-side cross-entropy (Cross-Ent), source-side cross-entropy difference (Moore-Lewis) from (Moore and Lewis, 2010), and bilingual cross-entropy difference (bML), which is novel.

Regardless of method, the overall procedure is the same. Using the scoring method, We rank the individual sentences of the general-domain corpus, select only the top N. We used the top $N = \{35k, 70k, 150k\}$ sentence pairs out of the 12 mil-

lion in the general corpus [5]. The net effect is that of domain adaptation via threshhold filtering. New MT systems were then trained solely on these small subcorpora, and compared against the baseline model trained on the entire 12m-sentence general-domain corpus. Table 2 contains BLEU scores of the systems trained on subsets of the general corpus.

Method	Sentences	Dev	Test
General	12m	42.62	40.51
Cross-Entropy	35k	39.77	40.66
Cross-Entropy	70k	40.61	**42.19**
Cross-Entropy	150k	42.73	**41.65**
Moore-Lewis	35k	36.86	40.08
Moore-Lewis	70k	40.33	39.07
Moore-Lewis	150k	41.40	40.17
bilingual M-L	35k	39.59	**42.31**
bilingual M-L	70k	40.84	**42.29**
bilingual M-L	150k	42.64	**42.22**

Table 2: Translation results using only a subset of the general-domain corpus.

All three methods presented for selecting a subset of the general-domain corpus (Cross-Entropy, Moore-Lewis, bilingual Moore-Lewis) could be used to train a state-of-the-art machine translation system. The simplest method, using only the source-side cross-entropy, was able to outperform the general-domain model when selecting 150k out of 12 million sentences. The other monolingual method, source-side cross-entropy difference, was able to perform nearly as well as the general-domain model with only 35k sentences. The bilingual Moore-Lewis method proposed in this paper works best, consistently boosting performance by 1.8 BLEU while using less than 1% of the available training data.

5.1 Pseudo In-Domain Data

The results in Table 2 show that all three methods (Cross-Entropy, Moore-Lewis, bilingual Moore-Lewis) can extract subsets of the general-domain corpus that are useful for the purposes of statistical machine translation. It is tempting to describe these as methods for finding in-domain data hidden in a general-domain corpus. Alas, this does not seem to be the case.

We trained a baseline language model on the in-domain data and used it to compute the perplexity of the same (in-domain) held-out dev set used to tune the translation models. We extracted the top N sentences using each ranking method, varying N from 10k to 200k, and then trained language models on these subcorpora. These were then used to also compute the perplexity of the same held-out dev set, shown below in Figure 1.

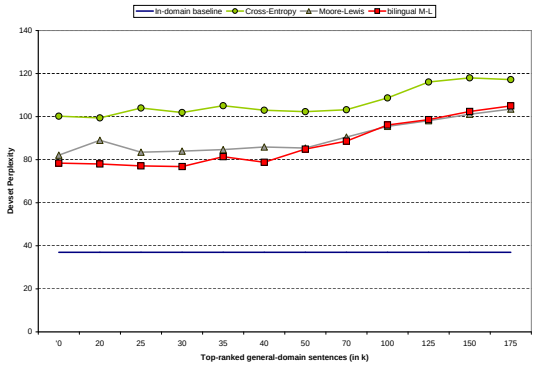


Figure 1: Corpus Selection Results

The perplexity of the dev set according to LMs trained on the top-ranked sentences varied from 77 to 120, depending on the size of the subset and the method used. The Cross-Entropy method was consistently worse than the others, with a best perplexity of 99.4 on 20k sentences, and bilingual Moore-Lewis was consistently the best, with a lowest perplexity of 76.8. And yet, none of these scores are anywhere near the perplexity of 36.96 according to the LM trained only on in-domain data.

From this it can be deduced that the selection methods are not finding data that is strictly in-domain. Rather they are extracting **pseudo in-domain** data which is relevant, but with a differing distribution than the original in-domain corpus.

As further evidence, consider the results of concatenating the in-domain corpus with the best extracted subcorpora (using the bilingual Moore-Lewis method), shown in Table 3. The change in

[5] Roughly 1x, 2x, and 4x the size of the in-domain corpus.

both the dev and test scores appears to reflect dissimilarity in the underlying data. Were the two datasets more alike, one would expect the models to reinforce each other rather than cancel out.

Method	Sentences	Dev	Test
IWSLT	30k	45.43	37.17
bilingual M-L	35k	39.59	**42.31**
bilingual M-L	70k	40.84	**42.29**
bilingual M-L	150k	42.64	**42.22**
IWSLT + bi M-L	35k	47.71	41.78
IWSLT + bi M-L	70k	47.80	**42.30**
IWSLT + bi M-L	150k	48.44	42.01

Table 3: Translation results concatenating the in-domain and pseudo in-domain data to train a single model.

6 Translation Model Combination

Because the pseudo in-domain data should be kept separate from the in-domain data, one must train multiple translation models in order to advantageously use the general-domain corpus. We now examine how best to combine these models.

6.1 Linear Interpolation

A common approach to managing multiple translation models is to interpolate them, as in (Foster and Kuhn, 2007) and (Lü et al., 2007). We tested the linear interpolation of the in- and general-domain translation models as follows: Given one model which assigns the probability $P_1(t|s)$ to the translation of source string s into target string t, and a second model which assigns the probability $P_2(t|s)$ to the same event, then the interpolated translation probability is:

$$P(t|s) = \lambda P_1(t|s) + (1 - \lambda)P_2(t|s) \qquad (4)$$

Here λ is a tunable weight between 0 and 1, which we tested in increments of 0.1. Linear interpolation of phrase tables was shown to improve performance over the individual models, but this still may not be the most effective use of the translation models.

6.2 Multiple Models

We next tested the approach in (Koehn and Schroeder, 2007), passing the two phrase tables directly to the decoder and tuning a system using both phrase tables in parallel. Each phrase table receives a separate set of weights during tuning, thus this combined translation model has more parameters than a normal single-table system.

Unlike (Nakov, 2008), we explicitly did not attempt to resolve any overlap between the phrase tables, as there is no need to do so with the multiple decoding paths. Any phrase pairs appearing in both models will be treated separately by the decoder. However, the exact overlap between the phrase tables was tiny, minimizing this effect.

6.3 Translation Model Combination Results

Table 4 shows baseline results for the in-domain translation system and the general-domain system, evaluated on the in-domain data. The table also shows that linearly interpolating the translation models improved the overall BLEU score, as expected. However, using multiple decoding paths, and no explicit model merging at all, produced even better results, by 2 BLEU points over the best individual model and 1.3 BLEU over the best interpolated model, which used $\lambda = 0.9$.

System	Dev	Test
IWSLT	45.43	37.17
General	42.62	40.51
Interpolate IWSLT, General	48.46	41.28
Use both IWSLT, General	49.13	**42.50**

Table 4: Translation model combination results

We conclude that it can be more effective to not attempt translation model adaptation directly, and instead let the decoder do the work.

7 Combining Multi-Model and Data Selection Approaches

We presented in Section 5 several methods to improve the performance of a single general-domain translation system by restricting its training corpus on an information-theoretic basis to a very small number of sentences. However, Section 6.3 shows that using two translation models over all the available data (one in-domain, one general-domain) outperforms any single individual translation model so far, albeit only slightly.

Method	Dev	Test
IWSLT	45.43	37.17
General	42.62	40.51
both IWSLT, General	49.13	42.50
IWSLT, Moore-Lewis 35k	48.51	40.38
IWSLT, Moore-Lewis 70k	49.65	40.45
IWSLT, Moore-Lewis 150k	49.50	41.40
IWSLT, bi M-L 35k	48.85	39.82
IWSLT, bi M-L 70k	49.10	**43.00**
IWSLT, bi M-L 150k	49.80	**43.23**

Table 5: Translation results from using in-domain and pseudo in-domain translation models together.

It is well and good to use the in-domain data to select pseudo in-domain data from the general-domain corpus, but given that this requires access to an in-domain corpus, one might as well use it. As such, we used the in-domain translation model alongside translation models trained on the subcorpora selected using the Moore-Lewis and bilingual Moore-Lewis methods in Section 4. The results are in Table 5.

A translation system trained on a pseudo in-domain subset of the general corpus, selected with the bilingual Moore-Lewis method, can be further improved by combining with an in-domain model. Furthermore, this system combination works better than the conventional multi-model approach by up to 0.7 BLEU on both the dev and test sets.

Thus a domain-adapted system comprising two phrase tables trained on a total of 180k sentences outperformed the standard multi-model system which was trained on 12 million sentences. This tiny combined system was also 3+ points better than the general-domain system by itself, and 6+ points better than the in-domain system alone.

8 Conclusions

Sentence pairs from a general-domain corpus that seem similar to an in-domain corpus may not actually represent the same distribution of language, as measured by language model perplexity. Nonetheless, we have shown that relatively tiny amounts of this *pseudo in-domain* data can prove more useful than the entire general-domain corpus for the purposes of domain-targeted translation tasks.

This paper has also explored three simple yet effective methods for extracting these pseudo in-domain sentences from a general-domain corpus. A translation model trained on any of these subcorpora can be comparable – or substantially better – than a translation system trained on the entire corpus.

In particular, the new bilingual Moore-Lewis method, which is specifically tailored to the machine translation scenario, is shown to be more efficient and stable for MT domain adaptation. Translation models trained on data selected in this way consistently outperformed the general-domain baseline while using as few as 35k out of 12 million sentences. This fast and simple technique for discarding over 99% of the general-domain training corpus resulted in an increase of 1.8 BLEU points.

We have also shown in passing that the linear interpolation of translation models may work less well for translation model adaptation than the multiple paths decoding technique of (Birch et al., 2007). These approaches of data selection and model combination can be stacked, resulting in a compact, two phrase-table, translation system trained on 1% of the available data that again outperforms a state-of-the-art translation system trained on all the data.

Besides improving translation performance, this work also provides a way to mine very large corpora in a computationally-limited environment, such as on an ordinary computer or perhaps a mobile device. The maximum size of a useful general-domain corpus is now limited only by the availability of data, rather than by how large a translation model can be fit into memory at once.

References

Amittai Axelrod. 2006. Factored Language Models for Statistical Machine Translation. *M.Sc. Thesis. University of Edinburgh, Scotland*.

Alexandra Birch, Miles Osborne and Philipp Koehn. 2007. CCG Supertags in Factored Translation Models. *Workshop on Statistical Machine Translation, Association for Computational Linguistics*.

Stanley Chen and Joshua Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. *Technical Report 10-98, Computer Science Group, Harvard University*.

Matthias Eck, Stephan Vogel, and Alex Waibel. 2004. Language Model Adaptation for Statistical Machine

Translation based on Information Retrieval. *Language Resources and Evaluation*.

George Foster and Roland Kuhn. 2007. Mixture-Model Adaptation for SMT. *Workshop on Statistical Machine Translation, Association for Computational Linguistics*.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative Instatnce Weighting for Domain Adaptation in Statistical Machine Translation. *Empirical Methods in Natural Language Processing*.

Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. 2002. Toward a Unified Approach to Statistical Language Modeling for Chinese. *ACM Transactions on Asian Language Information Processing*.

Xiaodong He. 2007. Using Word-Dependent Transition Models in HMM-based Word Alignment for Statistical Machine Translation. *Workshop on Statistical Machine Translation, Association for Computational Linguistics*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2003. Moses: Open Source Toolkit for Statistical Machine Translation. *Demo Session, Association for Computational Linguistics*.

Philipp Koehn and Josh Schroeder. 2007. Experiments in Domain Adaptation for Statistical Machine Translation. *Workshop on Statistical Machine Translation, Association for Computational Linguistics*.

Yajuan Lü, Jin Huang and Qun Liu. 2007. Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. *Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Spyros Matsoukas, Antti-Veikko Rosti, Bing Zhang. 2009. Discriminative Corpus Weight Estimation for Machine Translation. *Empirical Methods in Natural Language Processing*.

Robert Moore and William Lewis. 2010. Intelligent Selection of Language Model Training Data. *Association for Computational Linguistics*.

Preslav Nakov. 2008. Improving English-Spanish Statistical Machine Translation: Experiments in Domain Adaptation, Sentence Paraphrasing, Tokenization, and Recasing. *Workshop on Statistical Machine Translation, Association for Computational Linguistics*.

Franz Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. *Association for Computational Linguistics*

Andreas Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. *Spoken Language Processing*.

Keiji Yasuda, Ruiqiang Zhang, Hirofumi Yamamoto, Eiichiro Sumita. 2008. Method of Selecting Training Data to Build a Compact and Efficient Translation Model. *International Joint Conference on Natural Language Processing*.

Language Models for Machine Translation: Original vs. Translated Texts

Gennadi Lembersky and **Noam Ordan** and **Shuly Wintner**
Department of Computer Science, University of Haifa, 31905 Haifa, Israel
glembers@campus.haifa.ac.il, noam.ordan@gmail.com, shuly@cs.haifa.ac.il

Abstract

We investigate the differences between language models compiled from original target-language texts and those compiled from texts manually translated to the target language. Corroborating established observations of Translation Studies, we demonstrate that the latter are significantly better predictors of translated sentences than the former, and hence fit the reference set better. Furthermore, translated texts yield better language models for statistical machine translation than original texts.

1 Introduction

Statistical machine translation (MT) uses large target language models (LMs) to improve the fluency of generated texts, and it is commonly assumed that for constructing language models, "more data is better data" (Brants and Xu, 2009). Not all data, however, are created the same. In this work we explore the differences between LMs compiled from texts originally written in the target language and LMs compiled from *translated* texts.

The motivation for our work stems from much research in Translation Studies that suggests that original texts are significantly different from translated ones in various aspects (Gellerstam, 1986). Recently, corpus-based computational analysis corroborated this observation, and Kurokawa et al. (2009) apply it to statistical machine translation, showing that for an English-to-French MT system, a *translation* model trained on an English-translated-to-French parallel corpus is better than one trained on French-translated-to-English texts. Our research question is whether a *language model* compiled from translated texts may similarly improve the results of machine translation.

We test this hypothesis on several translation tasks, where the target language is always English. For each language pair we build two English language models from two types of corpora: texts originally written in English, and human translations from the source language into English. We show that for each language pair, the latter language model better fits a set of reference translations in terms of perplexity. We also demonstrate that the differences between the two LMs are not biased by content but rather reflect differences on abstract linguistic features.

Research in Translation Studies suggests that all translated texts, irrespective of source language, share some so-called *translation universals*. Consequently, translated texts from several languages to a single target language resemble each other along various axes. To test this hypothesis, we compile additional English LMs, this time using texts translated to English from languages *other* than the source. Again, we use perplexity to assess the fit of these LMs to reference sets of translated-to-English sentences. We show that these LMs depend on the source language and differ from each other. Whereas they outperform original-based LMs, LMs compiled from texts that were translated from the *source* language still fit the reference set best.

Finally, we train phrase-based MT systems (Koehn et al., 2003) for each language pair. We use four types of LMs: original; translated from

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 363–374,
Edinburgh, Scotland, UK, July 27–31, 2011.

the source language; translated from other languages; and a mixture of translations from several languages. We show that the translated-from-source-language LMs provide a significant improvement in the quality of the translation output over all other LMs, and that the mixture LMs always outperform the original LMs. This improvement persists even when the original LMs are up to ten times larger than the translated ones.

The main contributions of this work are therefore a computational corroboration of the hypotheses that

1. original and translated texts exhibit significant, measurable differences;
2. LMs compiled from translated texts better fit translated references than LMs compiled from original texts of the same (and much larger) size (and, to a lesser extent, LMs compiled from texts translated from languages other than the source language); and
3. MT systems that use LMs based on manually translated texts significantly outperform LMs based on originally written texts.

It is important to emphasize that translated texts abound: Many languages, especially low-resource ones, are more likely to have translated texts (religious scripts, educational materials, etc.) than original ones. Some numeric data are listed in Pym and Chrupała (2005). Furthermore, such data can be automatically identified (see Section 2). The practical impact of our work on MT is therefore potentially dramatic.

This paper is organized as follows: Section 2 provides background and describes related work. We explain our research methodology and resources in Section 3 and detail our experiments and results in Section 4. Section 5 discusses the results and their implications.

2 Background and Related Work

Numerous studies suggest that translated texts are different from original ones. Gellerstam (1986) compares texts written originally in Swedish and texts translated from English into Swedish. He notes that the differences between them do not indicate poor translation but rather a statistical phenomenon, which he terms *translationese*. He focuses mainly on lexical differences, for example less colloquialism in the translations, or foreign words used in the translations "with new shades of meaning taken from the English lexeme" (p.91). Only later studies consider grammatical differences (see, e.g., Santos (1995)). The features of translationese were theoretically organized under the terms *laws of translation* and *translation universals*.

Toury (1980, 1995) distinguishes between two laws: the *law of interference* and the *law of growing standardization*. The former pertains to the fingerprints of the source text that are left in the translation product. The latter pertains to the effort to standardize the translation product according to existing norms in the target language (and culture). Interestingly, these two laws are in fact reflected in the architecture of statistical machine translation: interference corresponds to the translation model and standardization to the language model.

The combined effect of these laws creates a hybrid text that partly corresponds to the source text and partly to texts written originally in the target language but in fact belongs to neither (Frawley, 1984). Baker (1993, 1995, 1996) suggests several candidates for translation universals, which are claimed to appear in any translated text, regardless of the source language. These include *simplification*, the tendency of translated texts to simplify the language, the message or both; and *explicitation*, their tendency to spell out implicit utterances that occur in the source text.

Baroni and Bernardini (2006) use machine learning techniques to distinguish between original and translated Italian texts, reporting 86.7% accuracy. They manage to abstract from content and perform the task using only morpho-syntactic cues. Ilisei et al. (2010) perform the same task for Spanish but enhance it theoretically in order to check the simplification hypothesis. The most informative features are lexical variety, sentence length and lexical density.

van Halteren (2008) focuses on six languages from Europarl (Koehn, 2005): Dutch, English, French, German, Italian and Spanish. For each

of these languages, a parallel six-lingual subcorpus is extracted, including an original text and its translations into the other five languages. The task is to identify the source language of translated texts, and the reported results are excellent. This finding is crucial: as Baker (1996) states, translations do resemble each other; however, in accordance with the law of interference, the study of van Halteren (2008) suggests that translation from different source languages constitute different sublanguages. As we show in Section 4.2, LMs based on translations from the source language outperform LMs compiled from non-source translations, in terms of both fitness to the reference set and improving MT.

Kurokawa et al. (2009) show that the direction of translation affects the performance of statistical MT. They train systems to translate between French and English (and vice versa) using a French-translated-to-English parallel corpus, and then an English-translated-to-French one. They find that in translating into French it is better to use the latter parallel corpus, and when translating into English it is better to use the former. Whereas they focus on the translation model, we focus on the language model in this work. We show that using a LM trained on a text translated from the source language of the MT system does indeed improve the results of the translation.

3 Methodology and Resources

3.1 Hypotheses

We investigate the following three hypotheses:

1. Translated texts differ from original texts;
2. Texts translated from one language differ from texts translated from other languages;
3. LMs compiled from manually translated texts are better for MT as measured using BLEU than LMs compiled from original texts.

We test our hypotheses by considering translations from several languages to English. For each language pair we create a reference set comprising several thousands of sentences written originally in the source language and manually translated to English. Section 3.4 provides details on the reference sets.

To investigate the first hypothesis, we train two LMs for each language pair, one created from original English texts and the other from texts translated into English. Then, we check which LM better fits the reference set.

Fitness of a LM to a set of sentences is measured in terms of *perplexity* (Jelinek et al., 1977; Bahl et al., 1983). Given a language model and a test (reference) set, perplexity measures the predictive power of the language model over the test set, by looking at the average probability the model assigns to the test data. Intuitively, a better model assigns higher probablity to the test data, and consequently has a *lower* perplexity; it is *less* surprised by the test data. Formally, the perplexity PP of a language model L on a test set $W = w_1\, w_2 \ldots w_N$ is the probability of W normalized by the number of words N Jurafsky and Martin (2008, page 96):

$$PP(L, W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P_L(w_i | w_1 \ldots w_{i-1})}} \quad (1)$$

For the second hypothesis, we extend the experiment to LMs created from texts translated from other languages to English. For example, we test how well a LM trained on French-to-English-translated texts fits the German-to-English reference set; and how well a LM trained on German-to-English-translated texts fits the French-to-English reference set.

Finally, for the third hypothesis, we use these LMs for statistical MT (SMT). For each language pair we build several SMT systems. All systems use a translation model extracted from a parallel corpus which is oblivious to the direction of the translation; and one of the above-mentioned LMs. Then, we compare the translation quality of these systems in terms of the BLEU metric (Papineni et al., 2002).

3.2 Language Models

In all the experiments, we use SRILM (Stolcke, 2002) to train 4-gram language models (with the default backoff model) from various corpora. Our main corpus is Europarl (Koehn, 2005), specifically portions collected over years 1996 to

1999 and 2001 to 2009. This is a large multilingual corpus, containing sentences translated from several European languages. However, it is organized as a collection of bilingual corpora rather than as a single multilingual one, and it is hard to identify sentences that are translated to several languages.

We therefore treat each bilingual sub-corpus in isolation; each such sub-corpus contains sentences translated from various languages. We rely on the **language** attribute of the **speaker** tag to identify the source language of sentences in the English part of the corpus. Since this tag is rarely used with English-language speakers, we also exploit the **ID** attribute of the **speaker** tag, which we match against the list of British members of the European parliament.

We focus on the following languages: German (DE), French (FR), Italian (IT), and Dutch (NL). For each of these languages, L, we consider the L-English Europarl sub-corpus. In each sub-corpus, we extract chunks of approximately 2.5 million *English* tokens translated from each of these source languages (T-L), as well as sentences written originally in English (O-EN). The mixture corpus (MIX), which is designed to represent "general" translated language, is constructed by randomly selecting sentences translated from any language (excluding original English sentences). Table 1 lists the number of sentences, number of tokens and average sentence length, for each sub-corpus and each original language.

In addition, we use the Hansard corpus, containing transcripts of the Canadian parliament from 1996–2007[1]. This is a bilingual French–English corpus comprising about 80% original English texts (EO) and about 20% texts translated from French (FO). We first separate original English from the original French and then, for each original language, we randomly extract portions of texts of different sizes: 1M, 5M and 10M tokens from the FO corpus and 1M, 5M, 10M, 25M, 50M and 100M tokens from the EO corpus; see Table 2.

[1]We are grateful to Cyril Goutte, George Foster and Pierre Isabelle for providing us with an annotated version of this corpus.

German–English			
Orig. Lang.	**Sent's**	**Tokens**	**Len**
MIX	82,700	2,325,261	28.1
O-EN	91,100	2,324,745	25.5
T-DE	87,900	2,322,973	26.4
T-FR	77,550	2,325,183	30.0
T-IT	65,199	2,325,996	35.7
T-NL	94,000	2,323,646	24.7
French–English			
Orig. Lang.	**Sent's**	**Tokens**	**Len**
MIX	90,700	2,546,274	28.1
O-EN	99,300	2,545,891	25.6
T-DE	94,900	2,546,124	26.8
T-FR	85,750	2,546,085	29.7
T-IT	72,008	2,546,984	35.4
T-NL	103,350	2,545,645	24.6
Italian–English			
Orig. Lang.	**Sent's**	**Tokens**	**Len**
MIX	87,040	2,534,793	29.1
O-EN	93,520	2,534,892	27.1
T-DE	90,550	2,534,867	28.0
T-FR	82,930	2,534,930	30.6
T-IT	69,270	2,535,225	36.6
T-NL	96,850	2,535,053	26.2
Dutch–English			
Orig. Lang.	**Sent's**	**Tokens**	**Len**
MIX	90,500	2,508,265	27.7
O-EN	97,000	2,475,652	25.5
T-DE	94,200	2,503,354	26.6
T-FR	86,600	2,523,055	29.1
T-IT	73,541	2,518,196	34.2
T-NL	101,950	2,513,769	24.7

Table 1: Europarl corpus statistics

To experiment with a non-European language (and a different genre) we choose Hebrew (HE). We use two English corpora: The *original* (O-EN) corpus comprises articles from the *International Herald Tribune*, downloaded over a period of seven months (from January to July 2009). The articles cover four topics: news (53.4%), business (20.9%), opinion (17.6%) and arts (8.1%). The *translated* (T-HE) corpus consists of articles collected from the Israeli newspaper *HaAretz* over the same period of time. *HaAretz* is published in Hebrew, but portions of

Original French			
Size	**Sent's**	**Tokens**	**Len**
1M	54,851	1,000,076	18.23
5M	276,187	5,009,157	18.14
10M	551,867	10,001,716	18.12
Original English			
Size	**Sent's**	**Tokens**	**Len**
1M	54,216	1,006,275	18.56
5M	268,806	5,006,482	18.62
10M	537,574	10,004,191	18.61
25M	1,344,580	25,001,555	18.59
50M	2,689,332	50,009,861	18.60
100M	5,376,886	100,016,704	18.60

Table 2: Hansard corpus statistics

it are translated to English. The O-corpus was downsized, so both corpora had approximately the same number of tokens in each topic. Table 3 lists basic statistics for these corpora.

Hebrew–English			
Orig. Lang.	**Sent's**	**Tokens**	**Len**
O-EN	135,228	3,561,559	26.3
T-HE	147,227	3,561,556	24.2

Table 3: Hebrew-to-English corpus statistics

3.3 SMT Training Data

To focus on the effect of the *language* model on translation quality, we design SMT training corpora to be oblivious to the direction of translation. Again, we use Europarl (January 2000 to September 2000) as the main source of our parallel corpora. We also use the Hansard corpus: We randomy extract 50,000 sentences from the original French sub-corpora and another 50,000 sentences from the original English sub-corpora. For Hebrew we use the Hebrew–English parallel corpus (Tsvetkov and Wintner, 2010) which contains sentences translated from Hebrew to English (54%) and from English to Hebrew (46%). The English-to-Hebrew part comprises many short sentences (approximately 6 tokens per sentence) taken from a movie subtitle database. This explains the small token to sentence ratio of this particular corpus. Table 4 lists some details on those corpora.

Lang's	**Side**	**Sent's**	**Tokens**	**Len**
DE-EN	DE	92,901	2,439,370	26.3
	EN	92,901	2,602,376	28.0
FR-EN	FR	93,162	2,610,551	28.0
	EN	93,162	2,869,328	30.8
IT-EN	IT	85,485	2,531,925	29.6
	EN	85,485	2,517,128	29.5
NL-EN	NL	84,811	2,327,601	27.4
	EN	84,811	2,303,846	27.2
Hansard	FR	100,000	2,167,546	21.7
	EN	100,000	1,844,415	18.4
HE-EN	HE	95,912	726,512	7.6
	EN	95,912	856,830	8.9

Table 4: SMT training data details

3.4 Reference Sets

The reference sets have two uses. First, they are used as the test sets in the experiments that measure the perplexity of the language models. Second, in the MT experiments we use them to randomly extract 1000 sentences for tuning and 1000 (different) sentences for evaluation.

For each language L we use the L-English subcorpus of Europarl (over the period of October to December 2000), containing only sentences originally produced in language L. The Hansard reference set is completely disjoint from the LM and SMT training sets and comprises only original French sentences. The Hebrew-to-English reference set is an independent (disjoint) part of the Hebrew-to-English parallel corpus. This set mostly comprises literary data (88.6%) and a small portion of news (11.4%). All sentences are originally written in Hebrew and are manually translated to English. See Table 5.

4 Experiments and Results

We detail in this section the experiments performed to test the three hypotheses: that translated texts can be distinguished from original ones, and provide better language models of other translated texts; that texts translated from other languages than the source are still better predictors of translations than original texts (Section 4.1); and that these differences are important for SMT (Section 4.2).

Lang's	Side	Sent's	Tokens	Len
DE-EN	DE	6,675	161,889	24.3
	EN	6,675	178,984	26.8
FR-EN	FR	8,494	260,198	30.6
	EN	8,494	271,536	32.0
IT-EN	IT	2,269	82,261	36.3
	EN	2,269	78,258	34.5
NL-EN	NL	4,593	114,272	24.9
	EN	4,593	105,083	22.9
Hansard	FR	8,926	193,840	21.72
	EN	8,926	163,448	18.3
HE-EN	HE	7,546	102,085	13.5
	EN	7,546	126,183	16.7

Table 5: Reference sets

4.1 Translated vs. Original texts

We train several 4-gram LMs for each Europarl sub-corpus, based on the corpora described in Section 3.2. For each language L, we train a LM based on texts translated from L, from languages other than L as well as texts originally written in English. The LMs are applied to the reference set of texts translated from L, and we compute the perplexity: the fitness of the LM to the reference set. Table 6 details the results, where for each sub-corpus and LM we list the number of unigrams in the test set, the number of out-of-vocabulary items (OOV) and the perplexity (PP). The lowest perplexity (reflecting the **best** fit) in each sub-corpus is typeset in boldface, and the highest (*worst* fit) is slanted.

These results overwhelmingly support our hypothesis. For each language L, the perplexity of the LM that was created from L translations is lowest, followed immediately by the MIX LM. Furthermore, the perplexity of the LM created from originally-English texts is highest in all experiments. In addition, the perplexity of LMs constructed from texts translated from languages other than L always lies between these two extremes: it is a better fit of the reference set than original texts, but not as good as texts translated from L (or mixture translations). This corroborates the hypothesis that translations form a language in itself, and translations from L_1 to L_2, form a sub-language, related to yet different from translations from other languages to L_2.

German to English translations			
Orig. Lang.	**Unigrams**	**OOV**	**PP**
MIX	32,238	961	83.45
O-EN	31,204	1161	*96.50*
T-DE	27,940	963	**77.77**
T-FR	29,405	1141	92.71
T-IT	28,586	1122	95.14
T-NL	28,074	1143	89.17
French to English translations			
Orig. Lang.	**Unigrams**	**OOV**	**PP**
MIX	33,444	1510	87.13
O-EN	32,576	1961	*105.93*
T-DE	28,935	2191	96.83
T-FR	30,609	1329	**82.23**
T-IT	29,633	1776	91.15
T-NL	29,221	2148	100.18
Italian to English translations			
Orig. Lang.	**Unigrams**	**OOV**	**PP**
MIX	33,353	462	90.71
O-EN	32,546	633	*107.45*
T-DE	28,835	628	100.46
T-FR	30,460	524	92.18
T-IT	29,466	470	**80.57**
T-NL	29,130	675	105.07
Dutch to English translations			
Orig. Lang.	**Unigrams**	**OOV**	**PP**
MIX	33,050	651	87.37
O-EN	32,064	771	*100.75*
T-DE	28,766	778	90.35
T-FR	30,502	775	96.38
T-IT	29,386	916	99.26
T-NL	29,178	560	**78.25**

Table 6: Fitness of various LMs to the reference set

A possible explanation for the different perplexity results between the LMs could be the specific contents of the corpora used to compile the LMs. To rule out this possibility and to further emphasize that the corpora are indeed *structurally* different, we conduct more experiments, in which we gradually abstract away from the domain- and content-specific features of the texts and emphasize their syntactic structure. We focus on German-to-English.

First, we remove all punctuation to eliminate

possible bias due to differences in punctuation conventions. Then, we use the Stanford Named Entity Recognizer (Finkel et al., 2005) to identify named entities, which we replace with a unique token ('NE'). Next, we replace all nouns with their POS tag; we use the Stanford POS Tagger (Toutanova and Manning, 2000). Finally, for full lexical abstraction, we replace all words with their POS tags.

At each step, we train six language models on O- and T-texts and apply them to the reference set (adapted to the same level of abstraction, of course). As the abstraction of the text increases, we also increase the order of the LMs: From 4-grams for text without punctuation and NE abstraction to 5-grams for noun abstraction to 8-grams for full POS abstraction. The results, which are depicted in Table 7, consistently show that the T-based LM is a better fit to the reference set, albeit to a lesser extent. While we do not show the details here, the same pattern is persistent in all the other Europarl languages we experiment with.

We repeat this experiment with the Hebrew-to-English reference set. We train two 4-gram LMs on the O-EN and T-HE corpora. We then apply the two LMs to the reference set and compute the perplexity. The results are presented in Table 8. Although the T-based LM has more OOVs, it is a better fit to the translated text than the O-based LM: Its perplexity is lower by 20.1%. Interestingly, the O-corpus LM has more unique unigrams than the T-corpus LM, supporting the claim of Al-Shabab (1996) that translated texts have lower type-to-token ratio.

We also conduct the above-mentioned abstraction experiments. The results, which are depicted in Table 9, consistently show that the T-based LM is a better fit to the reference set.

Clearly, then, translated LMs better fit the references than original ones, and the differences can be traced back not just to (trivial) specific lexical choice, but also to syntactic structure, as evidenced by the POS abstraction experiments. In fact, in order to retain the low perplexity level of translated texts, a LM based on original texts must be approximately ten times larger. We establish this by experimenting with the Hansard corpus. The results are persistent, but are omitted for lack of space.

No Punctuation			
Orig. Lang.	**OOVs**	**PP**	**PP diff.**
MIX	770	109.36	7.58%
O-EN	946	*127.03*	20.43%
T-DE	795	**101.07**	0.00%
T-FR	909	122.03	17.18%
T-IT	991	125.36	19.38%
T-NL	936	117.37	13.89%
NE Abstraction			
Orig. Lang.	**OOVs**	**PP**	**PP diff.**
MIX	643	99.13	6.99%
O-EN	772	*114.19*	19.26%
T-DE	661	**92.20**	0.00%
T-FR	752	110.22	16.35%
T-IT	823	112.72	18.21%
T-NL	771	105.81	12.86%
Noun Abstraction			
Orig. Lang.	**OOVs**	**PP**	**PP diff.**
MIX	400	38.48	4.71%
O-EN	459	*42.06*	12.80%
T-DE	405	**36.67**	0.00%
T-FR	472	40.96	10.47%
T-IT	489	41.39	11.39%
T-NL	440	39.54	7.26%
POS Abstraction			
Orig. Lang.	**OOVs**	**PP**	**PP diff.**
MIX	0	8.02	1.22%
O-EN	0	*8.19*	3.31%
T-DE	0	**7.92**	0.00%
T-FR	0	8.10	2.16%
T-IT	0	8.12	2.50%
T-NL	0	8.03	1.42%

Table 7: Fitness of O- vs. T-based LMs to the reference set (DE-EN), different abstraction levels

4.2 Original vs. Translated LMs for MT

The last hypothesis we test is whether a better fitting language model yields a better machine translation system. In other words, we expect the T-based LMs to outperform the O-based LMs when used as part of an MT system. We construct German-to-English, French-to-English, Italian-to-English and Dutch-to-

Hebrew to English translations			
Orig. Lang.	**Unigrams**	**OOV**	**PP**
O-EN	74,305	2,955	*282.75*
T-HE	61,729	3,253	**226.02**

Table 8: Fitness of O- vs. T-based LMs to the reference set (HE-EN)

No Punctuation			
Orig. Lang.	**OOVs**	**PP**	**PP diff.**
O-EN	2,601	442.95	19.2%
T-HE	2,922	358.11	0.0%
NE Abstraction			
Orig. Lang.	**OOVs**	**PP**	**PP diff.**
O-EN	1,794	350.3	17.3%
T-HE	2,038	289.71	0.0%
Noun Abstraction			
Orig. Lang.	**OOVs**	**PP**	**PP diff.**
O-EN	679	93.31	12.4%
T-HE	802	81.72	0.0%
POS Abstraction			
Orig. Lang.	**OOVs**	**PP**	**PP diff.**
O-EN	0	11.47	6.2%
T-HE	0	10.76	0.0%

Table 9: Fitness of O- vs. T-based LMs to the reference set (HE-EN), different abstraction levels

English MT systems using the Moses phrase-based SMT toolkit (Koehn et al., 2007). The systems are trained on the parallel corpora described in Section 3.3. We use the reference sets (Section 3.4) as follows: 1,000 sentences are randomly extracted for minimum error-rate tuning (Och, 2003), and another set of 1,000 sentences is randomly used for evaluation. Each system is built and tuned with six different LMs: MIX, O-based and four T-based (Section 3.2). We use BLEU (Papineni et al., 2002) to evaluate translation quality. The results are listed in Table 10.

These results are consistent: the translated-from-source systems outperform all other systems; mixture models come second; and systems that use original English LMs always perform worst. We test the statistical significance of differences between various MT systems using the bootstrap resampling method (Koehn, 2004). In all experiments, the best system (translated-from-source LM) is significantly better than all

DE to EN	
LM	**BLEU**
MIX	21.95
O-EN	*21.35*
T-DE	**22.42**
T-FR	21.47
T-IT	21.79
T-NL	21.59

IT to EN	
LM	**BLEU**
MIX	26.79
O-EN	*25.69*
T-DE	25.86
T-FR	26.56
T-IT	**27.28**
T-NL	25.77

FR to EN	
LM	**BLEU**
MIX	25.43
O-EN	*24.85*
T-DE	25.03
T-FR	**25.91**
T-IT	25.44
T-NL	25.17

NL to EN	
LM	**BLEU**
MIX	25.17
O-EN	*24.46*
T-DE	25.12
T-FR	24.79
T-IT	24.93
T-NL	**25.73**

Table 10: Machine translation with various LMs

other systems ($p < 0.05$); (even more) significantly better than the O-EN system ($p < 0.01$); and the mixture systems are significantly better than the O-EN systems ($p < 0.01$).

We also construct a Hebrew-to-English MT system using Moses' factored translation model (Koehn and Hoang, 2007). Every token in the training corpus is represented as two factors: surface form and lemma. Moreover, the Hebrew input is fully segmented. The system is built and tuned with O- and T-based LMs. Table 11 depicts the performance of the systems. The T-based LM yields a statistically better BLEU score than the O-based system.

LM	**BLEU**	**p-value**
O-based LM	*11.98*	0.012
T-based LM	**12.57**	

Table 11: Hebrew-to-English MT results

The LMs used in the above experiments are small. We now want to assess whether the benefits of using translated LMs carry over to scenarios where large original corpora exist. We build yet another set of French-to-English MT systems. We use the Hansard SMT translation model and Hansard LMs to train nine MT systems, three with varying sizes of translated texts and six with varying sizes of original texts.

We tune and evaluate on the Hansard reference set. In another set of experiments we use the Europarl French-to-English scenario (using Europarl corpora for the translation model and for tuning and evaluation), but we use the nine Hansard LMs to see whether our findings are consistent also when LMs are trained on out-of-domain (but similar genre) material.

Table 12 shows that the original English LMs should be enlarged by a factor of *ten* to achieve translation quality similar to that of translation-based LMs. In other words, much smaller translated LMs perform better than much larger original ones, and this is true for various LM sizes.

In-Domain		**Out-of-Domain**	
Original French		**Original French**	
Size	**BLEU**	**Size**	**BLEU**
1M	34.05	1M	18.87
5M	35.12	5M	23.90
10M	35.65	10M	24.36
Original English		**Original English**	
Size	**BLEU**	**Size**	**BLEU**
1M	32.57	1M	18.68
5M	33.37	5M	23.02
10M	33.92	10M	23.45
25M	34.71	25M	23.82
50M	34.85	50M	23.95
100M	35.36	100M	24.16

Table 12: The effect of LM size on MT performance

5 Discussion

We use language models computed from different types of corpora to investigate whether their fitness to a reference set of translated-to-English sentences can differentiate between them (and, hence, between the corpora on which they are based). Our main findings are that LMs compiled from manually translated corpora are much better predictors of translated texts than LMs compiled from original-language corpora of the same size. The results are robust, and are sustainable even when the corpora and the reference sentences are abstracted in ways that retain their syntactic structure but ignore specific word meanings. Furthermore, we show that translated LMs are better predictors of translated sentences even when the LMs are compiled from texts translated from languages *other* than the source language. However, LMs based on texts translated from the source language still outperform LMs translated from other languages.

We also show that MT systems based on translated-from-source-language LMs outperform MT systems based on originals LMs or LMs translated from other languages. Again, these results are robust and the improvements are statistically significant. This effect seems to be amplified as translation quality improves. Furthermore, our results show that original LMs require ten times more data to exhibit the same fitness to the reference set and the same translation quality as translated LMs.

More generally, this study confirms that insights drawn from the field of theoretical translation studies, namely the dual claim according to which (1) translations as such differ from originals, and (2) translations from different source languages differ from each other, can be verified experimentally and contribute to the performance of machine translation.

Future research is needed in order to understand *why* this is the case. One plausible hypothesis is that recurrent multiword expressions in the source language are frequently solved by human translations and each of these expressions converges to a set of high-quality translation equivalents which are represented in the LM. Another hypothesis is that since translation-based LMs represent a simplified mode of language use, the error potential is smaller. We therefore expect translation-based LMs to use more unmarked forms.

This work also bears on language typology: we conjecture that LMs compiled from texts translated not from the original language, but from a closely related one, can be better than texts translated from a more distant language. Some of our results support this hypothesis, but more research is needed in order to establish it.

Acknowledgements

This research was supported by the Israel Science Foundation (grant No. 137/06). We are grateful to Alon Lavie for his consistent help.

References

Omar S. Al-Shabab. *Interpretation and the language of translation: creativity and conventions in translation*. Janus, Edinburgh, 1996.

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190, 1983.

Mona Baker. Corpus linguistics and translation studies: Implications and applications. In Gill Francis Mona Baker and Elena Tognini-Bonelli, editors, *Text and technology: in honour of John Sinclair*, pages 233–252. John Benjamins, Amsterdam, 1993.

Mona Baker. Corpora in translation studies: An overview and some suggestions for future research. *Target*, 7(2):223–243, September 1995.

Mona Baker. Corpus-based translation studies: The challenges that lie ahead. In Gill Francis Mona Baker and Elena Tognini-Bonelli, editors, *Terminology, LSP and Translation. Studies in language engineering in honour of Juan C. Sager*, pages 175–186. John Benjamins, Amsterdam, 1996.

Marco Baroni and Silvia Bernardini. A new approach to the study of Translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259–274, September 2006. URL `http://llc.oxfordjournals.org/cgi/content/short/21/3/259?rss=1`.

Thorsten Brants and Peng Xu. Distributed language models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*, pages 3–4, Boulder, Colorado, May 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N/N09/N09-4002`.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219885.

William Frawley. Prolegomenon to a theory of translation. In William Frawley, editor, *Translation. Literary, Linguistic and Philosophical Perspectives*, pages 159–175. University of Delaware Press, Newark, 1984.

Martin Gellerstam. Translationese in Swedish novels translated from English. In Lars Wollin and Hans Lindquist, editors, *Translation Studies in Scandinavia*, pages 88–95. CWK Gleerup, Lund, 1986.

Iustina Ilisei, Diana Inkpen, Gloria Corpas Pastor, and Ruslan Mitkov. Identification of translationese: A machine learning approach. In Alexander F. Gelbukh, editor, *Proceedings of CICLing-2010: 11th International Conference on Computational Linguistics and Intelligent Text Processing*, volume 6008 of *Lecture Notes in Computer Science*, pages 503–511. Springer, 2010. ISBN 978-3-642-12115-9. URL `http://dx.doi.org/10.1007/978-3-642-12116-6`.

Frederick Jelinek, Robert L. Mercer, Lalit R. Bahl, and J. K. Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *Journal of the Acoustical Society of America*, 62:S63, November 1977. Supplement 1.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition, February 2008. ISBN 013122798X. URL `http://www.worldcat.org/isbn/013122798X`.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Philipp Koehn. Europarl: A parallel corpus for

statistical machine translation. MT Summit, 2005.

Philipp Koehn and Hieu Hoang. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D/D07/D07-1091`.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54. Association for Computational Linguistics, 2003.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-2045`.

David Kurokawa, Cyril Goutte, and Pierre Isabelle. Automatic detection of translated text and its impact on machine translation. In *Proceedings of MT-Summit XII*, 2009.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1075096.1075117.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA, 2002. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073083.1073135.

Anthony Pym and Grzegorz Chrupała. The quantitative analysis of translation flows in the age of an international language. In Albert Branchadell and Lovell M. West, editors, *Less Translated Languages*, pages 27–38. John Benjamins, Amsterdam, 2005.

Diana Santos. On grammatical translationese. In *In Koskenniemi, Kimmo (comp.), Short papers presented at the Tenth Scandinavian Conference on Computational Linguistics (Helsinki*, pages 29–30, 1995.

Andreas Stolcke. SRILM—an extensible language modeling toolkit. In *Procedings of International Conference on Spoken Language Processing*, pages 901–904, 2002. URL `citeseer.ist.psu.edu/stolcke02srilm.html`.

Gideon Toury. *In Search of a Theory of Translation*. The Porter Institute for Poetics and Semiotics, Tel Aviv University, Tel Aviv, 1980.

Gideon Toury. *Descriptive Translation Studies and beyond*. John Benjamins, Amsterdam / Philadelphia, 1995.

Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora*, pages 63–70, Morristown, NJ, USA, 2000. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1117794.1117802.

Yulia Tsvetkov and Shuly Wintner. Automatic acquisition of parallel corpora from websites with dynamic content. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*,

pages 3389–3392. European Language Resources Association (ELRA), May 2010. ISBN 2-9517408-6-7.

Hans van Halteren. Source language markers in EUROPARL translations. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 937–944, Morristown, NJ, USA, 2008. Association for Computational Linguistics. ISBN 978-1-905593-44-6.

Better Evaluation Metrics Lead to Better Machine Translation

Chang Liu[1] and **Daniel Dahlmeier**[2] and **Hwee Tou Ng**[1,2]
[1]Department of Computer Science, National University of Singapore
[2]NUS Graduate School for Integrative Sciences and Engineering
{liuchan1,danielhe,nght}@comp.nus.edu.sg

Abstract

Many machine translation evaluation metrics have been proposed after the seminal BLEU metric, and many among them have been found to consistently outperform BLEU, demonstrated by their better correlations with human judgment. It has long been the hope that by tuning machine translation systems against these new generation metrics, advances in automatic machine translation evaluation can lead directly to advances in automatic machine translation. However, to date there has been no unambiguous report that these new metrics can improve a state-of-the-art machine translation system over its BLEU-tuned baseline.

In this paper, we demonstrate that tuning Joshua, a hierarchical phrase-based statistical machine translation system, with the TESLA metrics results in significantly better human-judged translation quality than the BLEU-tuned baseline. TESLA-M in particular is simple and performs well in practice on large datasets. We release all our implementation under an open source license. It is our hope that this work will encourage the machine translation community to finally move away from BLEU as the unquestioned default and to consider the new generation metrics when tuning their systems.

1 Introduction

The dominant framework of machine translation (MT) today is statistical machine translation (SMT) (Hutchins, 2007). At the core of the system is the decoder, which performs the actual translation. The decoder is parameterized, and estimating the optimal set of parameter values is of paramount importance in getting good translations. In SMT, the parameter space is explored by a tuning algorithm, typically MERT (Minimum Error Rate Training) (Och, 2003), though the exact method is not important for our purpose. The tuning algorithm carries out repeated experiments with different decoder parameter values over a *development data set*, for which reference translations are given. An automatic MT evaluation metric compares the output of the decoder against the reference(s), and guides the tuning algorithm towards iteratively better decoder parameters and output translations. The quality of the automatic MT evaluation metric therefore has an immediate effect on the whole system.

The first automatic MT evaluation metric to show a high correlation with human judgment is BLEU (Papineni et al., 2002). Together with its close variant the NIST metric, they have quickly become the standard way of tuning statistical machine translation systems. While BLEU is an impressively simple and effective metric, recent evaluations have shown that many new generation metrics can outperform BLEU in terms of correlation with human judgment (Callison-Burch et al., 2009; Callison-Burch et al., 2010). Some of these new metrics include METEOR (Banerjee and Lavie, 2005; Lavie and Agarwal, 2007), TER (Snover et al., 2006), MAXSIM (Chan and Ng, 2008; Chan and Ng, 2009), and TESLA (Liu et al., 2010).

Given the close relationship between automatic MT and automatic MT evaluation, the logical expectation is that a better MT evaluation metric would

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 375–384,
Edinburgh, Scotland, UK, July 27–31, 2011.

lead to better MT systems. However, this linkage has not yet been realized. In the SMT community, MT tuning still uses BLEU almost exclusively.

Some researchers have investigated the use of better metrics for MT tuning, with mixed results. Most notably, Padó et al. (2009) reported improved human judgment using their entailment-based metric. However, the metric is heavy weight and slow in practice, with an estimated runtime of 40 days on the NIST MT 2002/2006/2008 dataset, and the authors had to resort to a two-phase MERT process with a reduced n-best list. As we shall see, our experiments use the similarly sized WMT 2010 dataset, and most of our runs take less than one day.

Cer et al. (2010) compared tuning a phrase-based SMT system with BLEU, NIST, METEOR, and TER, and concluded that BLEU and NIST are still the best choices for MT tuning, despite the proven higher correlation of METEOR and TER with human judgment.

In this work, we investigate the effect of MERT using BLEU, TER, and two variants of TESLA, TESLA-M and TESLA-F, on Joshua (Li et al., 2009), a state-of-the-art hierarchical phrase-based SMT system (Chiang, 2005; Chiang, 2007). Our empirical study is carried out in the context of WMT 2010, for the French-English, Spanish-English, and German-English machine translation tasks. We show that Joshua responds well to the change of evaluation metric, in that a system trained on metric M typically does well when judged by the same metric M. We further evaluate the different systems with manual judgments and show that the TESLA family of metrics (both TESLA-M and TESLA-F) significantly outperforms BLEU when used to guide the MERT search.

The rest of this paper is organized as follows. In Section 2, we describe the four evaluation metrics used. Section 3 outlines our experimental set up using the WMT 2010 machine translation tasks. Section 4 presents the evaluation results, both automatic and manual. Finally, we discuss our findings in Section 5, future work in Section 6, and conclude in Section 7.

2 Evaluation metrics

This section describes the metrics used in our experiments. We do not seek to explain all their variants and intricate details, but rather to outline their core characteristics and to highlight their similarities and differences. In particular, since all our experiments are based on single references, we omit the complications due to multiple references and refer our readers instead to the respective original papers for the details.

2.1 BLEU

BLEU is fundamentally based on n-gram match precisions. Given a reference text R and a translation candidate T, we generate the bag of all n-grams contained in R and T for $n = 1, 2, 3, 4$, and denote them as $\mathrm{BNG}^{\mathrm{n}}_{\mathrm{R}}$ and $\mathrm{BNG}^{\mathrm{n}}_{\mathrm{T}}$ respectively. The n-gram precision is thus defined as

$$P_n = \frac{|\mathrm{BNG}^{\mathrm{n}}_{\mathrm{R}} \cap \mathrm{BNG}^{\mathrm{n}}_{\mathrm{T}}|}{|\mathrm{BNG}^{\mathrm{n}}_{\mathrm{T}}|}$$

To compensate for the lack of the recall measure, and hence the tendency to produce short translations, BLEU introduces a *brevity penalty*, defined as

$$\mathrm{BP} = \begin{cases} 1 & \text{if} |\mathrm{T}| > |\mathrm{R}| \\ e^{1-|R|/|T|} & \text{if} |\mathrm{T}| \leq |\mathrm{R}| \end{cases}$$

where the $|\cdot|$ operator denotes the size of a bag or the number of words in a text. The metric is finally defined as

$$\mathrm{BLEU}(\mathrm{R}, \mathrm{T}) = \mathrm{BP} \times \sqrt[4]{P_1 P_2 P_3 P_4}$$

BLEU is a very simple metric requiring neither training nor language-specific resources. Its use of the brevity penalty is however questionable, as subsequent research on n-gram-based metrics has consistently found that recall is in fact a more potent indicator than precision (Banerjee and Lavie, 2005; Zhou et al., 2006; Chan and Ng, 2009). As we shall see, despite the BP term, BLEU still exhibits a strong tendency to produce short translations.

2.2 TER

TER is based on counting transformations rather than n-gram matches. The metric is defined as the

minimum number of edits needed to change a candidate translation T to the reference R, normalized by the length of the reference, i.e.,

$$\mathrm{TER(R,T)} = \frac{\text{number of edits}}{|\mathrm{R}|}$$

One edit is defined as one insertion, deletion, or substitution of a single word, or the shift of a contiguous sequence of words, regardless of size and distance. Minimizing the edit distance so defined has been shown to be NP-complete, so the evaluation is carried out in practice by a heuristic greedy search algorithm.

TER is a strong contender as the leading new generation automatic metric and has been used in major evaluation campaigns such as GALE. Like BLEU, it is simple and requires no language specific resources. TER also corresponds well to the human intuition of an evaluation metric.

2.3 TESLA-M

TESLA[1] is a family of linear programming-based metrics proposed by Liu et al. (2010) that incorporates many newer ideas. The simplest variation is TESLA-M[2], based on matching bags of n-grams (BNG) like BLEU. However, unlike BLEU, TESLA-M formulates the matching process as a real-valued linear programming problem, thereby allowing the use of weights. An example weighted BNG matching problem is shown in Figure 1.

Two kinds of weights are used in TESLA-M. First, the metric emphasizes the content words by discounting the weight of an n-gram by 0.1 for every function word it contains. Second, the *similarity* between two n-grams is a function dependent on the lemmas, the WordNet synsets (Fellbaum, 1998), and the POS tag of every word in the n-grams.

Each node in Figure 1 represents one weighted n-gram. The four in the top row represent one BNG, and the three at the bottom represent the other BNG. The goal of the linear programming problem is to assign weights to the links between the two BNGs, so as to maximize the sum of the products of the link weights and their corresponding similarity scores.

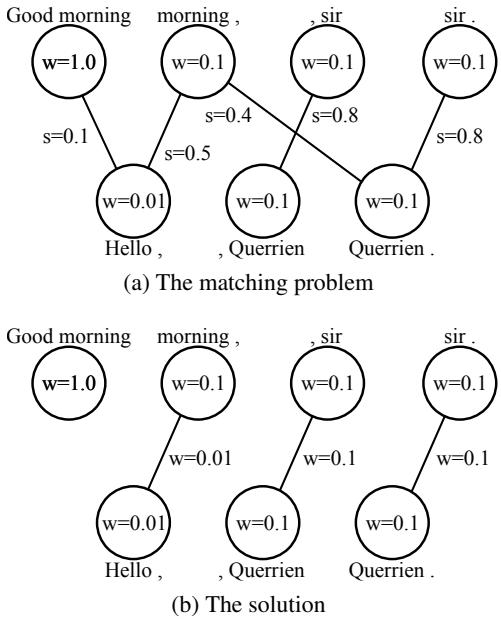


Figure 1: Matching two weighted bags of n-grams. w denotes the weight and s denotes the similarity.

The constraints of the linear programming problem are: (1) all assigned weights must be non-negative, and (2) the sum of weights assigned to all links connecting a node cannot exceed the node's weight. Intuitively, we allow splitting n-grams into fractional counts, and match them giving priority to the pairs with the highest similarities.

The linear programming formulation ensures that the matching can be solved uniquely and efficiently. Once the solution is found and let the maximized objective function value be S, the precision is computed as S over the sum of weights of the translation candidate n-grams. Similarly, the recall is S over the sum of weights of the reference n-grams. The precision and the recall are then combined to form the F-0.8 measure:

$$F_n = \frac{\text{Precision} \times \text{Recall}}{0.8 \times \text{Precision} + 0.2 \times \text{Recall}}$$

This F-measure gives more importance to the recall, reflecting its closer correlation with human judgment. F_n for $n = 1, 2, 3$ are calculated and averaged to produce the final score.

TESLA-M gains an edge over the previous two metrics by the use of lightweight linguistic features such as lemmas, synonym dictionaries, and POS

[1]The source code of TESLA is available at nlp.comp.nus.edu.sg/software/

[2]M stands for *minimal*.

Metric	Spearman's rho
TESLA-F	.94
TESLA-M	.93
meteor-next-*	.92
1-TERp	.90
BLEU-4-v13a-c	.89

Table 1: Selected system-level Spearman's rho correlation with the human judgment for the into-English task, as reported in WMT 2010.

Metric	Spearman's rho
TESLA-M	.93
meteor-next-rank	.82
1-TERp	.81
BLEU-4-v13a-c	.80
TESLA-F	.76

Table 2: Selected system-level Spearman's rho correlation with the human judgment for the out-of-English task, as reported in WMT 2010.

tags. While such tools are usually available even for languages other than English, it does make TESLA-M more troublesome to port to non-English languages.

TESLA-M did well in the WMT 2010 evaluation campaign. According to the system-level correlation with human judgments (Tables 1 and 2), it ranks top for the out-of-English task and very close to the top for the into-English task (Callison-Burch et al., 2010).

2.4 TESLA-F[3]

TESLA-F builds on top of TESLA-M. While word-level synonyms are handled in TESLA-M by examining WordNet synsets, no modeling of phrase-level synonyms is possible. TESLA-F attempts to remedy this shortcoming by exploiting a phrase table between the target language and another language, known as the pivot language.

Assume the target language is English and the pivot language is French, i.e., we are provided with an English-French phrase table. Let R and T be the

[3]TESLA-F refers to the metric called TESLA in (Liu et al., 2010). To minimize confusion, in this work we call the metric TESLA-F and refer to the whole family of metrics as TESLA. F stands for *full.*

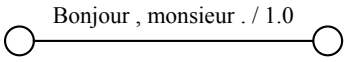


Figure 2: A degenerate confusion network in French. The phrase table maps *Good morning , sir .* to *Bonjour , monsieur .*

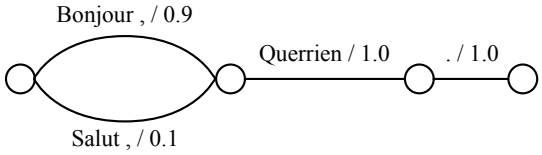


Figure 3: A confusion network in French. The phrase table maps *Hello ,* to *Bonjour ,* with $P = 0.9$ and to *Salut ,* with $P = 0.1$.

reference and the translation candidate respectively, both in English. As an example,

R: Good morning , sir .
T: Hello , Querrien .

TESLA-F first segments both R and T into phrases to maximize the probability of the sentences. For example, suppose both *Good morning , sir .* and *Hello ,* can be found in the English-French phrase table, and proper name *Querrien* is out-of-vocabulary, then a likely segmentation is:

R: ||| Good morning , sir . |||
T: ||| Hello , ||| Querrien ||| . |||

Each English phrase is then mapped to a bag of weighted French phrases using the phrase table, transforming the English sentences into confusion networks resembling Figures 2 and 3. French n-grams are extracted from these confusion network representations, known as pivot language n-grams. The bag of pivot language n-grams generated by R is then matched against that generated by T with the same linear programming formulation used in TESLA-M.

TESLA-F incorporates all the F-measures used in TESLA-M, with the addition of (1) the F-measures generated over the pivot language n-grams described above, and (2) the normalized language model score, defined as $\frac{1}{n} \log P$, where n is the length of the translation, and P the language model probability. Unlike BLEU and TESLA-M which rely on simple averages (geometric and arithmetic average respectively) to combine the component scores, TESLA-

F trains the weights over a set of human judgments using a linear ranking support vector machine (RSVM). This allows TESLA-F to exploit its components more effectively, but also makes it more tedious to work with and introduces potential domain mismatch problems.

TESLA-F makes use of even more linguistic information than TESLA-M, and has the capability of recognizing some forms of phrase synonyms. TESLA-F ranked top for the into-English evaluation task in WMT 2010 (Table 1). However, the added complexity, in particular the use of the language model score and the tuning of the component weights appear to make it less stable than TESLA-M in practice. For example, it did not perform as well in the out-of-English task.

3 Experimental setup

We run our experiments in the setting of the WMT 2010 news commentary machine translation campaign, for three language pairs:

1. French-English (fr-en): the training text consists of 84624 sentences of French-English bitext. The average French sentence length is 25 words.

2. Spanish-English (es-en): the training text consists of 98598 sentences of Spanish-English bitext. The average Spanish sentence length is 25 words.

3. German-English (de-en): the training text consists of 100269 sentences of German-English bitext. The average German sentence length is 22 words.

The average English sentence length is 21 words for all three language pairs. The text domain is newswire report, and the English sides of the training texts for the three language pairs overlap substantially. The development data are 2525 four-way translated sentences, in English, French, Spanish, and German respectively. Similarly, the test data are 2489 four-way translated sentences. As a consequence, all MT evaluations involve only single references.

We follow the standard approach for training hierarchical phrase-based SMT systems. First, we tokenize and lowercase the training texts and create word alignments using the Berkeley aligner (Liang et al., 2006; Haghighi et al., 2009) with five iterations of training. Then, we create suffix arrays and extract translation grammars for the development and test set with Joshua in its default setting. The maximum phrase length is 10. For the language model, we use SRILM (Stolcke, 2002) to build a trigram model with modified Kneser-Ney smoothing from the monolingual training data supplied in WMT 2010.

	fr-en	es-en	de-en
BLEU	3:49 (4)	5:09 (6)	2:41 (4)
TER	4:03 (4)	3:59 (4)	3:59 (5)
TESLA-M	13:00 (3)	17:34 (5)	13:40 (4)
TESLA-F	35:07 (4)	40:54 (4)	40:28 (5)

Table 3: Z-MERT training times in hours:minutes and number of iterations in parenthesis

Parameter tuning is carried out using Z-MERT (Zaidan, 2009). TER and BLEU are already implemented in the publicly released version of Z-MERT, and Z-MERT's modular design makes it easy to integrate TESLA-M and TESLA-F into the package. The maximum number of MERT iterations is set to 100, although we observe that in practice, the algorithm converges after 3 to 6 iterations. The number of intermediate initial points per iteration is set to 20 and the n-best list is capped to 300 translations. Table 3 shows the training times and the number of MERT iterations for each of the language pairs and evaluation metrics.

We use the publicly available version of TESLA-F, which comes with phrase tables and a ranking SVM model trained on the WMT 2010 development data.

4 Automatic and manual evaluations

The results of the automatic evaluations are presented in Table 4. The best score according to each metric is shown in bold. Note that smaller TER scores are better, as are larger BLEU, TESLA-M, and TESLA-F scores.[4]

We note that Joshua generally responds well to the change of tuning metric. A system tuned on met-

[4]The TESLA-F scores shown here have been monotonically scaled.

tune\test	BLEU	TER	TESLA-M	TESLA-F
BLEU	0.5237	0.6029	0.3922	0.4114
TER	**0.5239**	**0.6028**	0.3880	0.4095
TESLA-M	0.5005	0.6359	**0.4170**	0.4223
TESLA-F	0.4992	0.6377	0.4164	**0.4224**

(a) The French-English task

tune\test	BLEU	TER	TESLA-M	TESLA-F
BLEU	0.5641	0.5764	0.4315	0.4328
TER	**0.5667**	**0.5725**	0.4204	0.4282
TESLA-M	0.5253	0.6246	**0.4511**	0.4398
TESLA-F	0.5331	0.6111	0.4498	**0.4409**

(b) The Spanish-English task

tune\test	BLEU	TER	TESLA-M	TESLA-F
BLEU	**0.4963**	**0.6329**	0.3369	0.3927
TER	**0.4963**	0.6355	0.3191	0.3851
TESLA-M	0.4557	0.7055	**0.3784**	**0.4070**
TESLA-F	0.4642	0.6888	0.3753	0.4068

(c) The German-English task

Table 4: Automatic evaluation scores

	P(A)	Kappa
French-English	0.6846	0.5269
Spanish-English	0.6124	0.4185
German-English	0.3973	0.0960

Table 5: Inter-annotator agreement

ric M usually does the best or very close to the best when evaluated by M. On the other hand, the differences between different systems can be substantial, especially between BLEU/TER and TESLA-M/TESLA-F.

In addition to the automatic evaluation, we enlisted twelve judges to manually evaluate the first 200 test sentences. Four judges are assigned to each of the three language pairs. For each test sentence, the judges are presented with the source sentence, the reference English translation, and the output from the four competing Joshua systems. The order of the translation candidates is randomized so that the judges will not see any patterns. The judges are instructed to rank the four candidates, and ties are allowed.

The inter-annotator agreement is reported in Table 5. We consider the judgment for a pair of system outputs as one data point. Let $P(A)$ be the proportion of times that the annotators agree, and $P(E)$ be the proportion of times that they would agree by chance. The Kappa coefficient is defined as

$$\text{Kappa} = \frac{\text{P(A)} - \text{P(E)}}{1 - \text{P(E)}}$$

In our experiments, each data point has three possible values: A is preferred, B is preferred, and no preference, hence $P(E) = 1/3$. Our Kappa is calculated in the same way as the WMT workshops (Callison-Burch et al., 2009; Callison-Burch et al., 2010).

Kappa coefficients between 0.4 and 0.6 are considered *moderate*, and our values are in line with those reported in the WMT 2010 translation campaign. The exception is the German-English pair, where the annotators only reach *slight* agreement. This might be caused by the lower quality of German to English translations compared to the other two language pairs.

	fr-en	es-en	de-en
BLEU	44.1%	33.8%	49.6%
TER	41.4%	34.4%	47.8%
TESLA-M	65.8%	49.5%	57.8%
TESLA-F	66.4%	53.8%	55.1%

Table 6: Percentage of times each system produces the best translation

Table 6 shows the proportion of times each system produces the best translation among the four. We observe that the rankings are largely consistent across different language pairs: Both TESLA-F and TESLA-M strongly outperform BLEU and TER. Note that the values in each column do not add up to 100%, since the candidate translations are often identical, and even a different translation can receive the same human judgment.

Table 7 shows our main result, the pairwise comparison between the four systems for each of the language pairs. Again the rankings consistently show that both TESLA-F and TESLA-M strongly outperform BLEU and TER. All differences are statistically significant under the Sign Test at $p = 0.01$, with the exception of TESLA-M vs TESLA-F in the French-English task, BLEU vs TER in the Spanish-English task, and TESLA-M vs TESLA-F and BLEU vs TER in the German-English task. The results provide strong evidence that tuning machine

A\B	BLEU	TER	TESLA-M	TESLA-F
BLEU	-	11.4% / 6.5%	29.1% / 52.1%	28.0% / 52.3%
TER	6.5% / 11.4%	-	28.6% / 54.5%	27.5% / 55.0%
TESLA-M	52.1% / 29.1%	54.5% / 28.6%	-	~~7.6% / 8.8%~~
TESLA-F	52.3% / 28.0%	55.0% / 27.5%	~~8.8% / 7.6%~~	-

(a) The French-English task. All differences are significant under the Sign Test at $p = 0.01$, except the strikeout TESLA-M vs TESLA-F.

A\B	BLEU	TER	TESLA-M	TESLA-F
BLEU	-	~~25.8% / 22.3%~~	31.0% / 50.6%	24.4% / 50.8%
TER	~~22.3% / 25.8%~~	-	31.9% / 51.0%	26.4% / 52.4%
TESLA-M	50.6% / 31.0%	51.0% / 31.9%	-	25.9% / 33.4%
TESLA-F	50.8% / 24.4%	52.4% / 26.4%	33.4% / 25.9%	-

(b) The Spanish-English task. All differences are significant under the Sign Test at $p = 0.01$, except the strikeout BLEU vs TER.

A\B	BLEU	TER	TESLA-M	TESLA-F
BLEU	-	~~21.8% / 18.4%~~	28.1% / 36.9%	27.3% / 35.3%
TER	~~18.4% / 21.8%~~	-	26.9% / 39.5%	27.3% / 37.5%
TESLA-M	36.9% / 28.1%	39.5% / 26.9%	-	~~24.3% / 21.3%~~
TESLA-F	35.3% / 27.3%	37.5% / 27.3%	~~21.3% / 24.3%~~	-

(c) The German-English task. All differences are significant under the Sign Test at $p = 0.01$, except the strikeout BLEU vs TER, and TESLA-M vs TESLA-F.

Table 7: Pairwise system comparisons. Each cell shows the proportion of time the system tuned on A is preferred over the system tuned on B, and the proportion of time the opposite happens. Notice that the upper right half of each table is the mirror image of the lower left half.

translation systems using the TESLA metrics leads to significantly better translation output.

5 Discussion

We examined the results manually, and found that the relationship between the types of mistakes each system makes and the characteristics of the corresponding metric to be intricate. We discuss our findings in this section.

First we observe that BLEU and TER tend to produce very similar translations, and so do TESLA-F and TESLA-M. Of the 2489 test sentences in the French-English task, BLEU and TER produced different translations for only 760 sentences, or 31%. Similarly, TESLA-F and TESLA-M gave different outputs for only 857 sentences, or 34%. In contrast, BLEU and TESLA-M gave different translations for 2248 sentences, or 90%. It is interesting to find that BLEU and TER should be so similar, considering that they are based on very different principles. As a metric, TESLA-M is certainly much more similar to BLEU than TER is, yet they behave very differently when used as a tuning metric.

We also observe that TESLA-F and TESLA-M tend to produce much longer sentences than do BLEU and TER. The average sentence lengths of the TESLA-F- and TESLA-M-tuned systems across all three language pairs are 26.5 and 26.6 words respectively, whereas those for BLEU and TER are only 22.4 and 21.7 words. Comparing the translations from the two groups, the tendency of BLEU and TER to pick shorter paraphrases and to drop function words is unmistakable, often to the detriment of the translation quality. Some typical examples from the French-English task are shown in Figure 4.

Interestingly, the human translations average only 22 words, so BLEU and TER translations are in fact much closer on average to the reference lengths, yet their translations often feel too short. In contrast, manual inspections reveal no tendency for TESLA-F and TESLA-M to produce overly long translations.

These observations suggest that the brevity penalty of BLEU is not aggressive enough. Neither is TER, which penalizes insertions and deletions equally. Interestingly, by placing much more emphasis on the recall, TESLA-M and TESLA-F produce translations that are statistically too long, but feel much more 'correct' lengthwise.

Another major difference between TESLA-M/TESLA-F and BLEU/TER is that the TESLAs heavily discount n-grams with function words. One might thus expect the TESLA-tuned systems to be less adept at function words; yet they translate them surprisingly well, as shown in Figure 4. One explanation is of course the sentence length effect we have discussed. Another reason may be that since the metric does not care much about function words, the language model is given more freedom to pick function words as it sees fit, without the fear of large penalties. Paradoxically, by reducing the weights of function words, we end up making better translations for them.

TER is the only metric that allows cheap block movements, regardless of size or distance. One might reasonably speculate that a TER-tuned system should be more prone to reordering phrases. However, we find no evidence that this is so.

The relative performance of TESLA-M vs TESLA-F is unsurprising. TESLA-F, being heavier and slower, produces somewhat better results than its minimalist counterpart, though the margin is far less pronounced than that between TESLA-M and the conventional BLEU and TER. Since extra resources including bitexts are needed in using TESLA-F, TESLA-M emerges as the MT evaluation metric of choice for tuning SMT systems.

6 Future work

We have presented empirical evidence that the TESLA metrics outperform BLEU for MT tuning in a hierarchical phrase-based SMT system. At the same time, some open questions remain unanswered. We intend to investigate them in our future work.

The work of (Cer et al., 2010) investigated the effect of tuning a phrase-based SMT system and found that of the MT evaluation metrics that they tried, none of them could outperform BLEU. We would like to verify whether TESLA tuning is still preferred over BLEU tuning in a phrase-based SMT system.

Based on our observations, it may be possible to improve the performance of BLEU-based tuning by (1) increasing the brevity penalty; (2) introducing

BLEU	in the future , americans want a phone *that* allow the user to . . .
TER	in the future , americans want a phone *that* allow the user to . . .
TESLA-M	in the future , *the* americans want a *cell* phone *, which* allow the user to . . .
TESLA-F	in the future , *the* americans want a phone *that* allow the user to . . .
BLEU	. . . also for interest on debt of the state . . .
TER	. . . also for interest on debt of the state . . .
TESLA-M	. . . also for *the* interest on debt of the state . . .
TESLA-F	. . . also for *the* interest on debt of the state . . .
BLEU	and it is *hardly* the end of carnival-like transfers .
TER	and it is *hardly* the end of carnival-like transfers .
TESLA-M	and it is *far from being* the end of *the* carnival-like transfers .
TESLA-F	and it is *far from being* the end of *the* carnival-like transfers .
BLEU	it is not certain that the state can act without money .
TER	it is not certain that the state can act without money .
TESLA-M	it is not certain that the state can act without *this* money .
TESLA-F	it is not certain that the state can act without *this* money .
BLEU	but the expense of a debt of the state . . .
TER	but the expense of a debt of the state . . .
TESLA-M	but *at* the expense of a *greater* debt of the state . . .
TESLA-F	but *at* the expense of a *great* debt of the state . . .

Figure 4: Comparison of selected translations from the French-English task

a recall measure and emphasizing it over precision; and/or (3) introducing function word discounting. In the ideal case, such a modified BLEU metric would deliver results similar to that of TESLA-M, yet with a runtime cost closer to BLEU. It would also make porting existing tuning code easier.

7 Conclusion

We demonstrate for the first time that a practical new generation MT evaluation metric can significantly improve the quality of automatic MT compared to BLEU, as measured by human judgment. We hope this work will encourage the MT research community to finally move away from BLEU and to consider tuning their systems with a new generation metric.

All the data, source code, and results reported in this work can be downloaded from our website at `http://nlp.comp.nus.edu.sg/software`.

Acknowledgments

This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 workshop on statistical machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar F. Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*.

Daniel Cer, Christopher D. Manning, and Daniel Jurafsky. 2010. The best lexical metric for phrase-based statistical MT system optimization. In *Human Language Technologies: The 2010 Annual Conference of*

the North American Chapter of the Association for Computational Linguistics.

Yee Seng Chan and Hwee Tou Ng. 2008. MaxSim: A maximum similarity metric for machine translation evaluation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

Yee Seng Chan and Hwee Tou Ng. 2009. MaxSim: performance and effects of translation fluency. *Machine Translation*, 23(2):157–168, September.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. The MIT press.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised ITG models. In *Proceedings of 47th Annual Meeting of the Association for Computational Linguistics and the 4th IJCNLP of the AFNLP*.

John W. Hutchins. 2007. Machine translation: A concise history. *Computer Aided Translation: Theory and Practice*.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N.G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. 2010. Tesla: Translation evaluation of sentences with linear-programming-based analysis. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Sebastian Padó, Daniel Cer, Michel Galley, Dan Jurafsky, and Christopher D. Manning. 2009. Measuring machine translation quality as semantic equivalence: A metric based on entailment features. *Machine Translation*, 23(2):181–193, August.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the Seventh Conference of the Association for Machine Translation in the Americas*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*.

Omar Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Liang Zhou, Chin-Yew Lin, and Eduard Hovy. 2006. Re-evaluating machine translation results with paraphrase support. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*.

Evaluating Dependency Parsing: Robust and Heuristics-Free Cross-Annotation Evaluation

Reut Tsarfaty
Uppsala University
Sweden

Joakim Nivre
Uppsala University
Sweden

Evelina Andersson
Uppsala University
Sweden

Abstract

Methods for evaluating dependency parsing using attachment scores are highly sensitive to representational variation between dependency treebanks, making cross-experimental evaluation opaque. This paper develops a robust procedure for cross-experimental evaluation, based on deterministic unification-based operations for harmonizing different representations and a refined notion of tree edit distance for evaluating parse hypotheses relative to multiple gold standards. We demonstrate that, for different conversions of the Penn Treebank into dependencies, performance trends that are observed for parsing results in isolation change or dissolve completely when parse hypotheses are normalized and brought into the same common ground.

1 Introduction

Data-driven dependency parsing has seen a considerable surge of interest in recent years. Dependency parsers have been tested on parsing sentences in English (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; McDonald et al., 2005) as well as many other languages (Nivre et al., 2007a). The evaluation metric traditionally associated with dependency parsing is based on scoring labeled or unlabeled attachment decisions, whereby each correctly identified pair of head-dependent words is counted towards the success of the parser (Buchholz and Marsi, 2006). As it turns out, however, such evaluation procedures are sensitive to the annotation choices in the data on which the parser was trained.

Different annotation schemes often make different assumptions with respect to how linguistic content is represented in a treebank (Rambow, 2010). The consequence of such annotation discrepancies is that when we compare parsing results across different experiments, even ones that use the same parser and the same set of sentences, the gap between results in different experiments may not reflect a true gap in performance, but rather a difference in the annotation decisions made in the respective treebanks.

Different methods have been proposed for making dependency parsing results comparable across experiments. These methods include picking a single gold standard for all experiments to which the parser output should be converted (Carroll et al., 1998; Cer et al., 2010), evaluating parsers by comparing their performance in an embedding task (Miyao et al., 2008; Buyko and Hahn, 2010), or neutralizing the arc direction in the native representation of dependency trees (Schwartz et al., 2011).

Each of these methods has its own drawbacks. Picking a single gold standard skews the results in favor of parsers which were trained on it. Transforming dependency trees to a set of pre-defined labeled dependencies, or into task-based features, requires the use of heuristic rules that run the risk of distorting correct information and introducing noise of their own. Neutralizing the direction of arcs is limited to unlabeled evaluation and local context, and thus may not cover all possible discrepancies.

This paper proposes a new three-step protocol for cross-experiment parser evaluation, and in particular for comparing parsing results across data sets that adhere to different annotation schemes. In the

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 385–396,
Edinburgh, Scotland, UK, July 27–31, 2011.

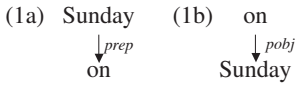
first step all structures are brought into a single formal space of events that neutralizes representation peculiarities (for instance, arc directionality). The second step formally computes, for each sentence in the data, the common denominator of the different gold standards, containing all and only linguistic content that is shared between the different schemes. The last step computes the normalized distance from this common denominator to parse hypotheses, minus the cost of distances that reflect mere annotation idiosyncrasies. The procedure that implements this protocol is fully deterministic and heuristics-free.

We use the proposed procedure to compare dependency parsing results trained on Penn Treebank trees converted into dependency trees according to five different sets of linguistic assumptions. We show that when starting off with the same set of sentences and the same parser, training on different conversion schemes yields apparently significant performance gaps. When results across schemes are normalized and compared against the shared linguistic content, these performance gaps decrease or dissolve completely. This effect is robust across parsing algorithms. We conclude that it is imperative that cross-experiment parse evaluation be a well thought-through endeavor, and suggest ways to extend the protocol to additional evaluation scenarios.

2 The Challenge: Treebank Theories

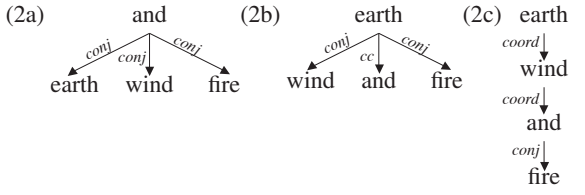
Dependency treebanks contain information about the grammatically meaningful elements in the utterance and the grammatical relations between them. Even if the formal representation in a dependency treebank is well-defined according to current standards (Kübler et al., 2009), there are different ways in which the trees can be used to express syntactic content (Rambow, 2010). Consider, for instance, algorithms for converting the phrase-structure trees in the Penn Treebank (Marcus et al., 1993) into dependency structures. Different conversion algorithms implicitly make different assumptions about how to represent linguistic content in the data. When multiple conversion algorithms are applied to the same data, we end up with different dependency trees for the same sentences (Johansson and Nugues, 2007; Choi and Palmer, 2010; de Marneffe et al., 2006). Some common cases of discrepancies are as follows.

Lexical vs. Functional Head Choice. In linguistics, there is a distinction between lexical heads and functional heads. A lexical head carries the semantic gist of a phrase while a functional one marks its relation to other parts of the sentence. The two kinds of heads may or may not coincide in a single word form (Zwicky, 1993). Common examples refer to prepositional phrases, such as the phrase "on Sunday". This phrase has two possible analyses, one selects a lexical head (1a) and the other selects a functional one (1b), as depicted below.



Similar choices are found in phrases which contain functional elements such as determiners, coordination markers, subordinating elements, and so on.

Multi-Headed Constructions. Some phrases are considered to have multiple lexical heads, for instance, coordinated structures. Since dependency-based formalisms require us to represent all content as binary relations, there are different ways we could represent such constructions. Let us consider the coordination of nominals below. We can choose between a functional head (1a) and a lexical head (2b, 2c). We can further choose between a flat representation in which the first conjunct is a single head (2b), or a nested structure where each conjunct/marker is the head of the following element (2c). All three alternatives empirically exist. Example (2a) reflects the structures in the CoNLL 2007 shared task data (Nivre et al., 2007a). Johansson and Nugues (2007) use structures like (2b). Example (2c) reflects the analysis of Mel'čuk (1988).



Periphrastic Marking. When a phrase includes periphrastic marking — such as the tense and modal marking in the phrase "would have worked" below — there are different ways to consider its division into phrases. One way to analyze this phrase would be to choose auxiliaries as heads, as in (3a). Another alternative would be to choose the final verb as the

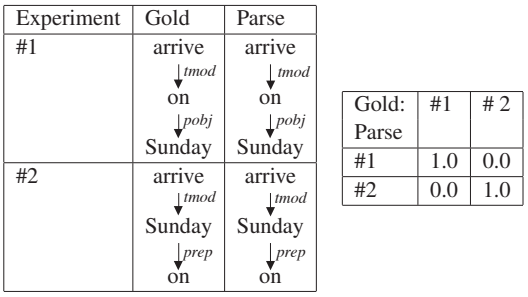
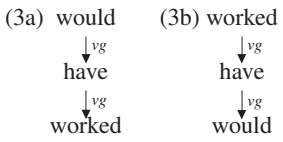


Figure 1: Calculating cross-experiment LAS results

main head, and let the auxiliaries create a verb chain with different levels of projection. Each annotation decision dictates a different direction of the arcs and imposes its own internal division into phrases.



In standard settings, an experiment that uses a data set which adheres to a certain annotation scheme reports results that are compared against the annotation standard that the parser was trained on. But if parsers were trained on different annotation standards, the empirical results are not comparable across experiments. Consider, for instance, the example in Figure 1. If parse1 and parse2 are compared against gold2 using labeled attachment scores (LAS), then parse1 results are lower than the results of parse2, even though both parsers produced linguistically correct and perfectly useful output.

Existing methods for making parsing results comparable across experiments include heuristics for converting outputs into dependency trees of a predefined standard (Briscoe et al., 2002; Cer et al., 2010) or evaluating the performance of a parser within an embedding task (Miyao et al., 2008; Buyko and Hahn, 2010). However, heuristic rules for cross-annotation conversion are typically hand written and error prone, and may not cover all possible discrepancies. Task-based evaluation may be sensitive to the particular implementation of the embedding task and the procedures that extract specific task-related features from the different parses. Beyond that, conversion heuristics and task-based procedures are currently developed almost exclusively for English. Other languages typically lack such resources.

A recent study by Schwartz et al. (2011) takes a different approach towards cross-annotation evaluation. They consider different directions of head-dependent relations (such as on→Sunday and Sunday→on) and different parent-child and grandparent-child relations in a chain (such as arrive→on and arrive→sunday in "arrive on sunday") as equivalent. They then score arcs that fall within corresponding equivalence sets. Using these new scores Schwartz et al. (2011) neutralize certain annotation discrepancies that distort parse comparison. However, their treatment is limited to local context and does not treat structures larger than two sequential arcs. Additionally, since arcs in different directions are typically labeled differently, this method only applies for unlabeled dependencies.

What we need is a fully deterministic and formally precise procedure for comparing any set of labeled or unlabeled dependency trees, by consolidating the shared linguistic content of the complete dependency trees in different annotation schemes, and comparing parse hypotheses through sound metrics that can take into account multiple gold standards.

3 The Proposal: Cross-Annotation Evaluation in Three Simple Steps

We propose a new protocol for cross-experiment parse evaluation, consisting of three fundamental components: (i) abstracting away from annotation peculiarities, (ii) generalizing theory-specific structures into a single linguistically coherent gold standard that contains all and only consistent information from all sources, and (iii) defining a sound metric that takes into account the different gold standards that are being considered in the experiments.

In this section we first define *functional trees* as the common space of formal objects and define a deterministic conversion procedure from dependency trees to functional trees. Next we define a set of formal operations on functional trees that compute, for every pair of corresponding trees of the same yield, a single gold tree that resolves inconsistencies among gold standard alternatives and combines the information that they share. Finally, we define scores based on *tree edit distance*, refined to consider the distance from parses to the overall gold tree as well as the different annotation alternatives.

Preliminaries. Let $\mathcal{T}$ be a finite set of terminal symbols and let $\mathcal{L}$ be a set of grammatical relation labels. A dependency graph d is a directed graph which consists of nodes V_d and arcs $A_d \subseteq V_d \times V_d$. We assume that all nodes in V_d are labeled by terminal symbols via a function $label_V : V_d \rightarrow \mathcal{T}$. A well-formed dependency graph $d = (V_d, A_d)$ for a sentence $S = t_1, t_2, ..., t_n$ is any dependency graph that is a directed tree originating out of a node v_0 labeled $t_0 = ROOT$, and spans all terminals in the sentence, that is, for every $t_i \in S$ there exists $v_j \in V_d$ labeled $label_V(v_j) = t_i$. For simplicity we assume that every node v_j is indexed according to the position of the terminal label, i.e., that for each t_i labeling v_j, i always equals j. In a labeled dependency tree, arcs in A_d are labeled by elements of $\mathcal{L}$ via a function $label_A : A_d \rightarrow \mathcal{L}$ that encodes the grammatical relation between the terminals labeling the connected nodes. We define two auxiliary functions on nodes in dependency trees. The function $subtree : V_d \rightarrow \mathcal{P}(V_d)$ assigns to every node $v \in V_d$ the set of nodes accessible by it through the reflexive transitive closure of the arc relation A_d. The function $span : V_d \rightarrow \mathcal{P}(\mathcal{T})$ assigns to every node $v \in V_d$ a set of terminals such that $span(v) = \{t \in \mathcal{T} | t = label_V(u) \text{ and } u \in subtree(v)\}$.[1]

Step 1: Functional Representation Our first goal is to define a representation format that keeps all functional relationships that are represented in the dependency trees intact, but remains neutral with respect to the directionality of the head-dependent relations. To do so we define *functional trees* — linearly-ordered labeled trees which, instead of head-to-head binary relations, represent the complete functional structure of a sentence. Assuming the same sets of terminal symbols $\mathcal{T}$ and grammatical relation labels $\mathcal{L}$, and assuming extended sets of nodes V and arcs $A \subseteq V \times V$, a functional tree $\pi = (V, A)$ is a directed tree originating from a single root $v_0 \in V$ where all non-terminal nodes in π are labeled with grammatical relation labels that signify the grammatical function of the chunk they dominate inside the tree via $label_{NT} : V \rightarrow \mathcal{L}$. All terminal nodes in π are labeled with terminal symbols via a $label_T : V \rightarrow \mathcal{T}$ function. The function $span : V \rightarrow \mathcal{P}(V)$ now picks out the set of terminal labels of the terminal nodes accessible by a node $v \in V$ via A. We obtain functional trees from dependency trees using the following procedure:

- Initialize the set of nodes and arcs in the tree.

  $V := V_d$
  $A := A_d$

- Label each node $v \in V$ with the label of its incoming arc.

  $label_{NT}(v) = label_A(u, v)$

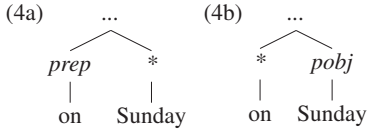
- In case $|span(v)| > 1$ add a new node u as a daughter designating the lexical head, labeled with the wildcard symbol *:

  $V := V \cup \{u\}$
  $A := A \cup \{(v, u)\}$
  $label_{NT}(u) = *$

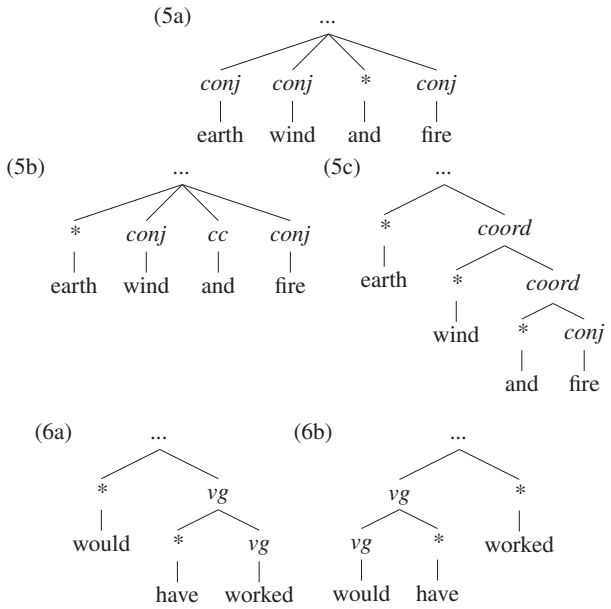
- For each node v such that $|span(v)| = 1$, add a new node u as a daughter, labeled with its own terminal:

  $V := V \cup \{u\}$
  $A := A \cup \{(v, u)\}$
  if $(label_{NT}(v) \neq *)$
  $label_T(u) := label_V(v)$
  else
  $label_T(u) := label_V(parent(v))$

That is to say, we label all nodes with spans greater than 1 with the grammatical function of their head, and for each node we add a new daughter u designating the head word, labeled with its grammatical function. Wildcard labels are compatible with any, more specific, grammatical function of the word inside the phrase. This gives us a constituency-like representation of dependency trees labeled with functional information, which retains the linguistic assumptions reflected in the dependency trees. When applying this procedure, examples (1)–(3) get transformed into (4)–(6) respectively.



[1] If a dependency tree d is projective, than for all $v \in V_d$ the terminals in $span(v)$ form a contiguous segment of S. The current discussion assumes that all trees are projective. We comment on non-projective dependencies in Section 4.



Considering the functional trees resulting from our procedure, it is easy to see that for tree pairs (4a)–(4b) and (5a)–(5b) the respective functional trees are identical modulo wildcards, while tree pairs (5b)–(5c) and (6a)–(6b) end up with different tree structures that realize different assumptions concerning the internal structure of the tree. In order to compare, combine or detect inconsistencies in the information inherent in different functional trees, we define a set of formal operations that are inspired by familiar notions from unification-based formalisms (Shieber (1986) and references therein).

Step 2: Formal Operations on Trees The intuition behind the formal operations we define is simple. A completely flat tree over a span is the most general structural description that can be given to it. The more nodes dominate a span, the more linguistic assumptions are made with respect to its structure. If an arc structure in one tree merely elaborates an existing flat span in another tree, the theories underlying the schemes are compatible, and their information can be combined. Otherwise, there exists a conflict in the linguistic assumptions, and we need to relax some of the assumptions, i.e., remove functional nodes, in order to obtain a coherent structure that contains the information on which they agree.

Let π_1, π_2 be functional trees over the same yield $t_1, .., t_n$. Let the function $span(v)$ pick out the terminals labeling terminal nodes that are accessible via a node $v \in V$ in the functional tree through the relation A. We define first the tree subsumption relation for comparing the amount of information inherent in the arc-structure of two trees.[2]

> **T-Subsumption**, denoted $\sqsubseteq_t$, is a relation between trees which indicates that a tree π_1 is consistent with and more general than tree π_2. Formally: $\pi_1 \sqsubseteq_t \pi_2$ iff for every node $n \in \pi_1$ there exists a node $m \in \pi_2$ such that $span(n) = span(m)$ and $label(n) = label(m)$.

Looking at the functional trees of (4a)–(4b) we see that their unlabeled skeletons mutually subsume each other. In their labeled versions, however, each tree contains labeling information that is lacking in the other. In the functional trees (5b)–(5c) a flat structure over a span in (5b) is more elaborated in (5c). In order to combine information in trees with compatible arc structures, we define tree unification.

> **T-Unification**, denoted $\sqcup_t$, is the operation that returns the most general tree structure π_3 that is subsumed by both π_1, π_2 if such exists, and fails otherwise. Formally: $\pi_1 \sqcup_t \pi_2 = \pi_3$ iff $\pi_1 \sqsubseteq_t \pi_3$ and $\pi_2 \sqsubseteq_t \pi_3$, and for all π_4 such that $\pi_1 \sqsubseteq_t \pi_4$ and $\pi_2 \sqsubseteq_t \pi_4$ it holds that $\pi_3 \sqsubseteq_t \pi_4$.

Tree unification collects the information from two trees into a single result if they are consistent, and detects an inconsistency otherwise. In case of an inconsistency, as is the case in the functional trees (6a) and (6b), we cannot unify the structures due to a conflict concerning the internal division of an expression into phrases. However, we still want to generalize these two trees into one tree that contains all and only the information that they share. For that we define the tree generalization operation.

> **T-Generalization**, denoted $\sqcap_t$, is the operation that returns the most specific tree that is more general than both trees. Formally, $\pi_1 \sqcap_t \pi_2 = \pi_3$ iff $\pi_3 \sqsubseteq_t \pi_1$ and $\pi_3 \sqsubseteq_t \pi_2$, and for every π_4 such that $\pi_4 \sqsubseteq_t \pi_1$ and $\pi_4 \sqsubseteq_t \pi_2$ it holds that $\pi_4 \sqsubseteq_t \pi_3$.

[2]Note that the wildcard symbol * is equal to any other symbol. In case the node labels consist of complex feature structures made of attribute-value lists, we replace $label(n) = label(m)$ in the subsumption definition with $label(n) \sqsubseteq label(m)$ in the sense of (Shieber, 1986).

Unlike unification, generalization can never fail. For every pair of trees there exists a tree that is more general than both: in the extreme case, pick the completely flat structure over the yield, which is more general than any other structure. For (6a)–(6b), for instance, we get that (6a)$\sqcap_t$(6b) is a flat tree over pre-terminals where "would" and "have" are labeled with *'vg'* and "worked" is the head, labeled with *'*'*.

The generalization of two functional trees provides us with one structure that reflects the common and consistent content of the two trees. These structures thus provide us with a formally well-defined gold standard for cross-treebank evaluation.

Step 3: Measuring Distances. Our functional trees superficially look like constituency-based trees, so a simple proposal would be to use Parseval measures (Black et al., 1991) for comparing the parsed trees against the new generalized gold trees. Parseval scores, however, have two significant drawbacks. First, they are known to be too restrictive with respect to some errors and too permissive with respect to others (Carroll et al., 1998; Kübler and Telljohann, 2002; Roark, 2002; Rehbein and van Genabith, 2007). Secondly, F_1 scores would still penalize structures that are correct with respect to the original gold, but are not there in the generalized structure. Here we propose to adopt measures that are based on tree edit distance (TED) instead. TED-based measures are, in fact, an extension of attachment scores for dependency trees. Consider, for instance, the following operations on dependency arcs.

> **reattach-arc** remove arc $(u, v) \in A_d$ and add an arc $A_d \cup \{(w, v)\}$.
>
> **relabel-arc** relabel arc $l_1(u, v)$ as $l_2(u, v)$

Assuming that each operation is assigned a cost, the attachment score of comparing two dependency trees is simply the cost of all edit operations that are required to turn a parse tree into its gold standard, normalized with respect to the overall size of the dependency tree and subtracted from a unity.[3] Here we apply the idea of defining scores by TED costs normalized relative to the size of the tree and substracted from a unity, and extend it from fixed-size dependency trees to ordered trees of arbitrary size.

[3]The size of a dependency tree, either parse or gold, is always fixed by the number of terminals.

Our formalization follows closely the formulation of the T-Dice measure of Emms (2008), building on his thorough investigation of the formal and empirical differences between TED-based measures and Parseval. We first define for any ordered and labeled tree π the following operations.

> **relabel-node** change the label of node v in π
>
> **delete-node** delete a non-root node v in π with parent u, making the children of v the children of u, inserted in the place of v as a subsequence in the left-to-right order of the children of u.
>
> **insert-node** insert a node v as a child of u in π making it the parent of a consecutive subsequence of the children of u.

An edit script $ES(\pi_1, \pi_2) = \{e_0, e_1....e_k\}$ between π_1 and π_2 is a set of edit operations required for turning π_1 into π_2. Now, assume that we are given a cost function defined for each edit operation. The cost of $ES(\pi_1, \pi_2)$ is the sum of the costs of the operations in the script. An optimal edit script is an edit script between π_1 and π_2 of minimum cost.

$$ES^*(\pi_1, \pi_2) = \text{argmin}_{ES(\pi_1,\pi_2)} \sum_{e \in ES(\pi_1,\pi_2)} cost(e)$$

The tree edit distance problem is defined to be the problem of finding the optimal edit script and computing the corresponding distance (Bille, 2005).

A simple way to calculate the error δ of a parse would be to define it as the edit distance between the parse hypothesis π_1 and the gold standard π_2.

$$\delta(\pi_1, \pi_2) = cost(ES^*(\pi_1, \pi_2))$$

However, in such cases the parser may still get penalized for recovering nodes that are lacking in the generalization. To solve this, we refine the distance between a parse tree and the generalized gold tree to discard edit operations on nodes that are there in the native gold tree but are eliminated through generalization. We compute the intersection of the edit script turning the parse tree into the generalize gold with the edit script turning the native gold tree into the generalized gold, and discard its cost. That is, if parse1 and parse2 are compared against gold1 and gold2 respectively, and if we set gold3 to be the result of gold1$\sqcap_t$gold2, then δ_{new} is defined as:

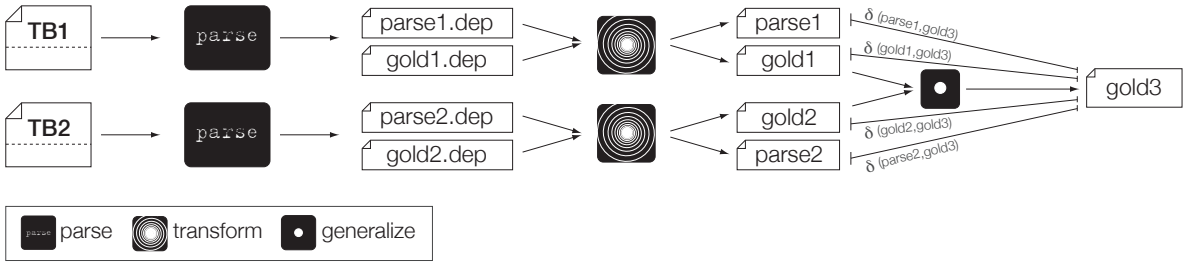


Figure 2: The evaluation pipeline. Different versions of the treebank go into different experiments, resulting in different parse and gold files. All trees are transformed into functional trees. All gold files enter generalization to yield a new gold. The different δ arcs represent the different tree distances used for calculating the TED-based scores.

δ_{new}(parse1, gold1,gold3) =
δ(parse1,gold3)
$-cost(ES^*$(parse1,gold3)$\cap ES^*$(gold1,gold3))

Now, if gold1 and gold3 are identical, then ES^*(gold1,gold3)=$\emptyset$ and we fall back on the simple tree edit distance score δ_{new}(parse1,gold1,gold3)=δ(parse1, gold3).
When parse1 and gold1 are identical, i.e., the parser produced perfect output with respect to its own scheme, then δ_{new}(parse1,gold1,gold3)=δ_{new}(gold1,gold1,gold3) =δ(gold1,gold3) $- cost(ES^*$(gold1,gold3))=0, and the parser does not get penalized for recovering a correct structure in gold1 that is lacking in gold3.

In order to turn distances into accuracy measures we have to normalize distances relative to the maximal number of operations that is conceivable. In the worst case, we would have to remove all the internal nodes in the parse tree and add all the internal nodes of the generalized gold, so our normalization factor ι is defined as follows, where $|\pi|$ is the size[4] of π.

$$\iota(\text{parse1,gold3}) = |\text{parse1}| + |\text{gold3}|$$

We now define the *score* of parse1 as follows:[5]

$$1 - \frac{\delta_{new}(\text{parse1,gold1,gold3})}{\iota(\text{parse1,gold3})}$$

Figure 2 summarizes the steps in the evaluation procedure we defined so far. We start off with two versions of the treebank, TB1 and TB2, which are parsed separately and provide their own gold standards and parse hypotheses in a labeled dependencies format. All dependency trees are then converted into functional trees, and we compute the generalization of each pair of gold trees for each sentence in the data. This provides the generalized gold standard for all experiments, here marked as gold3.[6] We finally compute the distances δ_{new}(parse1,gold1,gold3) and δ_{new}(parse2,gold2,gold3) using the different tree edit distances that are now available, and we repeat the procedure for each sentence in the test set.

To normalize the scores for an entire test set of size n we can take the arithmetic mean of the scores.

$$\frac{\sum_{i=1}^{|\text{test-set}|} score(\text{parse1}_i,\text{gold1}_i,\text{gold3}_i)}{|\text{test-set}|}$$

Alternatively we can globally average of all edit distance costs, normalized by the maximally possible edits on parse trees turned into generalized trees.

$$1 - \frac{\sum_{i=1}^{|\text{test-set}|} \delta_{new}(\text{parse1}_i,\text{gold1}_i,\text{gold3}_i)}{\sum_{i=1}^{|\text{test-set}|} \iota(\text{parse1}_i,\text{gold3}_i)}$$

The latter score, global averaging over the entire test set, is the metric we use in our evaluation procedure.

4 Experiments

We demonstrate the application of our procedure to comparing dependency parsing results on different versions of the Penn Treebank (Marcus et al., 1993).

The Data We use data from the PTB, converted into dependency structures using the LTH software, a general purpose tool for constituency-to-dependency conversion (Johansson and Nugues, 2007). We use LTH to implement the five different annotation standards detailed in Table 3.

[4]Following common practice, we equate size $|\pi|$ with the number of nodes in π, discarding the terminals and root node.

[5]If the trees have only root and leaves, $\iota = 0$, $score := 1$.

[6]Generalization is an associative and commutative operation, so it can be extended for n experiments in any order.

Train Gold		Default	Old LTH	CoNLL07
Default	UAS	**0.9142**	0.6077	0.7772
	LAS	**0.8820**	0.4801	0.6454
	U-TED	**0.9488**	0.8926	0.9237
	L-TED	**0.9241**	0.7811	0.8441
Old LTH	UAS	0.6053	**0.8955**	0.6508
	LAS	0.4816	**0.8644**	0.5771
	U-TED	0.8931	**0.9564**	0.9092
	L-TED	0.7811	**0.9317**	0.8197
CoNLL07	UAS	0.7734	0.6474	**0.8917**
	LAS	0.6479	0.5722	**0.8736**
	U-TED	0.9260	0.9097	**0.9474**
	L-TED	0.8480	0.8204	**0.9233**
Default-OldLTH	U-TED	0.9500	**0.9543**	
	L-TED	0.9278	**0.9324**	
Default-CoNLL07	U-TED	0.9444†		**0.9453**†
	L-TED	**0.9266**†		0.9260†
oldLTH-CoNLL07	U-TED		**0.9519**	0.9490
	L-TED		**0.9323**	0.9283
default-oldLTH-CoNLL	U-TED	0.9464†	**0.9515**	0.9471†
	L-TED	0.9281†	**0.9336**	0.9280†

Train Gold		CoNLL07	Functional	Lexical
CoNLL07	UAS	**0.8917**	0.8054	0.6986
	LAS	**0.8736**	0.7895	0.6831
	U-TED	**0.9474**	0.9357	0.9237
	L-TED	**0.9233**	0.8960	0.8606
Functional	UAS	0.8040	**0.8970**	0.6110
	LAS	0.7873	**0.8793**	0.5977
	U-TED	0.9347	**0.9466**	0.9107
	L-TED	0.8948	**0.9239**	0.8316
Lexical	UAS	0.7013	0.6138	**0.8823**
	LAS	0.6875	0.6022	**0.8635**
	U-TED	0.9252	0.9132	**0.9500**
	L-TED	0.8623	0.8345	**0.9266**
CoNLL07-Functional	U-TED	0.9473†	0.9473†	
	L-TED	0.9233	**0.9247**	
CoNLL07-Lexical	U-TED	0.9490†		**0.9500**†
	L-TED	0.9253†		**0.9266**†
Functional-Lexical	U-TED		0.9489†	**0.9501**†
	L-TED		0.9266†	**0.9267**†
CoNLL07-Functional-Lexical	U-TED	0.9489†	0.9489†	**0.9501**†
	L-TED	0.9254†	0.9266†	**0.9267**†

Table 1: Cross-experiment dependency parsing evaluation for MaltParser trained on multiple schemes. We report standard LAS scores and TEDEVAL global average metrics. Boldface results outperform the rest of the results reported in the same row. The † sign marks pairwise results where the difference is not statistically significant.

Train Gold		Default	Old LTH	CoNLL07
Default	UAS	**0.9173**	0.6085	0.7709
	LAS	**0.8833**	0.4780	0.6414
	U-TED	**0.9513**	0.8903	0.9236
	L-TED	**0.9249**	0.7727	0.8424
Old LTH	UAS	0.6078	**0.8952**	0.6415
	LAS	0.4809	**0.8471**	0.5669
	U-TED	0.8960	**0.9550**	0.9096
	L-TED	0.7823	**0.9224**	0.8170
CoNLL07	UAS	0.7767	0.6517	**0.8991**
	LAS	0.6504	0.5725	**0.8709**
	U-TED	0.9289	0.9087	**0.9479**
	L-TED	0.8502	0.8159	**0.9208**
Default-oldLTH	U-TED	**0.9533**	0.9515	
	L-TED	**0.9289**	0.9224	
Default-CoNLL	U-TED	**0.9474**†		0.9460†
	L-TED	**0.9281**		0.9238
OldLTH-CoNLL	U-TED		0.9479	**0.9493**
	L-TED		0.9234	**0.9258**
Default-OldLTH-CoNLL	U-TED	**0.9492**†	0.9461	0.9480†
	L-TED	**0.9298**	0.9241†	0.9258†

Train Gold		CoNLL07	Functional	Lexical
CoNLL07	UAS	**0.8991**	0.8077	0.7018
	LAS	**0.8709**	0.7902	0.6804
	U-TED	**0.9479**	0.9373	0.9221
	L-TED	**0.9208**	0.8955	0.8505
Functional	UAS	0.8083	**0.8978**	0.6150
	LAS	0.7895	**0.8782**	0.5975
	U-TED	0.9356	**0.9476**	0.9092
	L-TED	0.8929	**0.9226**	0.8218
Lexical	UAS	0.6997	0.6161	**0.8826**
	LAS	0.6835	0.6034	**0.8491**
	U-TED	0.9259	0.9152	**0.9483**
	L-TED	0.8593	0.8340	**0.9160**
CoNLL-Functional	U-TED	0.9479†	**0.9487**†	
	L-TED	0.9209	**0.9237**	
CoNLL-Lexical	U-TED	**0.9497**		0.9483
	L-TED	**0.9228**		0.9161
Functional-Lexical	U-TED		**0.9504**	0.9483
	L-TED		**0.9258**	0.9161
CoNLL-Functional-Lexical	U-TED	0.9498	**0.9504**†	0.9483†
	L-TED	0.9229	**0.9258**	0.9161

Table 2: Cross-experiment dependency parsing evaluation for the MST parser trained on multiple schemes. We report standard LAS scores and TEDEVAL global average metrics. Boldface results outperform the rest of the results reported in the same row. The † sign marks pairwise results where the difference is not statistically significant.

ID	Description
Default	The LTH conversion default settings
OldLTH	The conversion used in Johansson and Nugues (2007)
CoNLL07	The conversion used in the CoNLL shared task (Nivre et al., 2007a)
Lexical	Same as **CoNLL**, but selecting only lexical heads when a choice exists
Functional	Same as **CoNLL**, but selecting only functional heads when a choice exists

Table 3: LTH conversion schemes used in the experiments. The LTH conversion settings in terms of the complete feature-value pairs associated with the LTH parameters in different schemes are detailed in the supplementary material.

The Default, OldLTH and CoNLL schemes mainly differ in their coordination structure, and the Functional and Lexical schemes differ in their selection of a functional and a lexical head, respectively. All schemes use the same inventory of labels.[7] The LTH parameter settings for the different schemes are elaborated in the supplementary material.

The Setup We use two different parsers: (i) MaltParser (Nivre et al., 2007b) with the arc eager algorithm as optimized for English in (Nivre et al., 2010) and (ii) MSTParser with the second-order projective model of McDonald and Pereira (2006). Both parsers were trained on the different instances of sections 2-21 of the PTB obeying the different annotation schemes in Table 3. Each trained model was used to parse section 23. All non-projective dependencies in the training and gold sets were projectivized prior to training and parsing using the algorithm of Nivre and Nilsson (2005). A more principled treatment of non-projective dependency trees is an important topic for future research. We evaluated the parses using labeled and unlabeled attachment scores, and using our TEDEVAL software package.

Evaluation Our TEDEVAL software package implements the pipeline described in Section 3. We convert all parse and gold trees into functional trees using the algorithm defined in Section 3, and for each pair of parsing experiments we calculate a shared gold standard using generalization determined through a chart-based greedy algorithm.[8] Our scoring procedure uses the TED algorithm defined by Zhang and Shasha (1989).[9] The unlabeled score is obtained by assigning $cost(e) = 0$ for every e relabeling operation. To calculate pairwise statistical significance we use a shuffling test with 10,000 iterations (Cohen, 1995). A sample of all files in the evaluation pipeline for a subset of 10 PTB sentences is available in the supplementary materials.[10]

[7] In case the labels are not taken from the same inventory, e.g., subjects in one scheme are marked as SUB and in the other marked as SBJ, it is possible define a a set of zero-cost operation types — in such case, to the operation **relabel**(SUB,SBJ) — in order not to penalize string label discrepancies.

[8] Our algorithm has space and runtime complexity of $O(n^2)$.

[9] Available via `http://web.science.mq.edu.au/~swan/howtos/treedistance/`

[10] The TEDEVAL software package is available via `http://stp.lingfil.uu.se/~tsarfaty/unipar`

Results Table 1 reports the results for the inter- and cross-experiment evaluation of parses produced by MaltParser. The left hand side of the table presents the parsing results for a set of experiments in which we compare parsing results trained on the Default, OldLTH and CoNLL07 schemes. In a second set of experiments we compare the CoNLL07, Lexical and Functional schemes. Table 2 reports the evaluation of the parses produced by MSTParser for the same experimental setup. Our goal here is not to compare the parsers, but to verify that the effects of switching from LAS to TEDEVAL are robust across parsing algorithms.

In each of the tables, the top three groups of four rows compare results of parsed dependency trees trained on a particular scheme against gold trees of the same and the other schemes. The next three groups of two rows report the results for comparing pairwise sets of experiments against a generalized gold using our proposed procedure. In the last group of two rows we compare all parsing results against a single gold obtained through a three-way generalization.

As expected, every parser appears to perform at its best when evaluated against the scheme it was trained on. This is the case for both LAS and TEDEVAL measures and the performance gaps are statistically significant. When moving to pairwise evaluation against a single generalized gold, for instance, when comparing CoNLL07 to the Default settings, there is still a gap in performance, e.g., between OldLTH and CoNLL07, and between OldLTH and Default. This gap is however a lot smaller and is not always statistically significant. In fact, when evaluating the effect of linguistically disparate annotation variations such as Lexical and Functional on the performance of MaltParser, Table 1 shows that when using TEDEVAL and a generalized gold the performance gaps are small and statistically insignificant.

Moreover, observed performance trends when evaluating individual experiments on their original training scheme may change when compared against a generalized gold. The Default scheme, for MaltParser, appears better than OldLTH when both are evaluated against their training schemes. But looking at the pairwise-evaluated experiments, it is the other way round (the difference is smaller, but statistically significant). In evaluating against a three-way

generalization, all the results obtained for different training schemes are on a par with one another, with minor gaps in performance, rarely statistically significant. This suggests that apparent performance trends between experiments when evaluating with respect to the training schemes may be misleading.

These observations are robust across parsing algorithms. In each of the tables, results obtained against the training schemes show significant differences whereas applying our cross-experimental procedure shows small to no gaps in performance across different schemes. Annotation variants which seem to have crucial effects have a relatively small influence when parsed structures are brought into the same formal and theoretical common ground for comparison. Of course, it may be the case that one parser is better trained on one scheme while the other utilizes better another scheme, but objective performance gaps can only be observed when they are compared against shared linguistic content.

5 Discussion and Extensions

This paper addresses the problem of cross-experiment evaluation. As it turns out, this problem arises in NLP in different shapes and forms; when evaluating a parser against different annotation schemes, when evaluating parsing performance across parsers and different formalisms, and when comparing parser performance across languages. We consider our contribution successful if after reading it the reader develops a healthy suspicion to blunt comparison of numbers across experiments, or better yet, across different papers. Cross-experiment comparison should be a careful and well thought-through endeavor, in which we retain as much information as we can from the parsed structures, avoid lossy conversions, and focus on an object of evaluation which is agreed upon by all variants.

Our proposal introduces one way of doing so in a streamlined, efficient and formally worked out way. While individual components may be further refined or improved, the proposed setup and implementation can be straightforwardly applied to cross-parser and cross-framework evaluation. In the future we plan to use this procedure for comparing constituency and dependency parsers. A conversion from constituency-based trees into functional trees is straightforward to define: simply replace the node labels with the grammatical function of their dominating arc – and the rest of the pipeline follows.

A pre-condition for cross-framework evaluation is that all representations encode the same set of grammatical relations by, e.g., annotating arcs in dependency trees or decorating nodes in constituency trees. For some treebanks this is already the case (Nivre and Megyesi, 2007; Skut et al., 1997; Hinrichs et al., 2004) while for others this is still lacking. Recent studies (Briscoe et al., 2002; de Marneffe et al., 2006) suggest that evaluation through a single set of grammatical relations as the common denominator is a linguistically sound and practically useful way to go. To guarantee extensions for cross-framework evaluation it would be fruitful to make sure that resources use the same set of grammatical relation labels across different formal representation types. Moreover, we further aim to inquire whether we can find a single set of grammatical relation labels that can be used across treebanks for multiple languages. This would then pave the way for the development of cross-language evaluation procedures.

6 Conclusion

We propose an end-to-end procedure for comparing dependency parsing results across experiments based on three steps: (i) converting dependency trees to functional trees, (ii) generalizing functional trees to harmonize information from different sources, and (iii) using distance-based metrics that take the different sources into account. When applied to parsing results of different dependency schemes, dramatic gaps observed when comparing parsing results obtained in isolation decrease or dissolve completely when using our proposed pipeline.

Acknowledgments We thank the developers of the LTH and TED software who made their code available for our use. We thank Richard Johansson for providing us with the LTH parameter settings of existing dependency schemes. We thank Ari Rappoport, Omri Abend, Roy Schwartz and members of the NLP lab at the Hebrew University of Jerusalem for stimulating discussion. We finally thank three anonymous reviewers for useful comments on an earlier draft. The research reported in the paper was partially funded by the Swedish Research Council.

References

Philip Bille. 2005. A survey on tree edit distance and related. problems. *Theoretical Computer Science*, 337:217–239.

Ezra Black, Steven P. Abney, D. Flickenger, Claudia Gdaniec, Ralph Grishman, P. Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith L. Klavans, Mark Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In E. Black, editor, *Proceedings of the workshop on Speech and Natural Language*, HLT, pages 306–311. Association for Computational Linguistics.

Ted Briscoe, John Carroll, Jonathan Graham, and Ann Copestake. 2002. Relational evaluation schemes. In *Proceedings of LREC Workshop"Beyond Parseval – Towards improved evaluation measures for parsing systems"*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164.

Ekaterina Buyko and Udo Hahn. 2010. Evaluating the impact of alternative dependency graph encodings on solving event extraction tasks. In *Proceedings of EMNLP*, pages 982–992.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of LREC*, pages 447–454.

Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of LREC*.

Jinho D. Choi and Martha Palmer. 2010. Robust constituent-to-dependency conversion for English. In *Proceedings of TLT*.

Paul Cohen. 1995. *Empirical Methods for Artificial Intelligence*. The MIT Press.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, pages 449–454.

Martin Emms. 2008. Tree-distance and some other variants of evalb. In *Proceedings of LREC*.

Erhard Hinrichs, Sandra Kübler, Karin Naumann, Heike Telljohan, and Julia Trushkina. 2004. Recent development in linguistic annotations of the TüBa-D/Z Treebank. In *Proceedings of TLT*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA*.

Sandra Kübler and Heike Telljohann. 2002. Towards a dependency-oriented evaluation for partial parsing. In *Proceedings of LREC Workshop"Beyond Parseval – Towards improved evaluation measures for parsing systems"*.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Number 2 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of ACL*, pages 46–54.

Joakim Nivre and Beata Megyesi. 2007. Bootstrapping a Swedish Treebank using cross-corpus harmonization and annotation projection. In *Proceedings of TLT*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo projective dependency parsing. In *Proceeding of ACL*, pages 99–106.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING*, pages 64–70.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

Joakim Nivre, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(1):1–41.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez-Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. pages 813–821.

Owen Rambow. 2010. The Simple Truth about Dependency and Phrase Structure Representations: An Opinion Piece. In *Proceedings of HLT-ACL*, pages 337–340.

Ines Rehbein and Josef van Genabith. 2007. Why is it so difficult to compare treebanks? Tiger and TüBa-D/Z revisited. In *Proceedings of TLT*, pages 115–126.

Brian Roark. 2002. Evaluating parser accuracy using edit distance. In *Proceedings of LREC Workshop "Beyond Parseval – Towards improved evaluation measures for parsing systems"*.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of ACL*, pages 663–672.

Stuart M. Shieber. 1986. *An Introduction to Unification-Based Grammars*. Center for the Study of Language and Information.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word-order languages. In *Proceedings of the fifth conference on Applied natural language processing*, pages 88–95.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceeding of IWPT*, pages 195–206.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. In *SIAM Journal of Computing*, volume 18, pages 1245–1262.

Arnold M. Zwicky. 1993. Heads, bases, and functors. In G.G. Corbett, N. Fraser, and S. McGlashan, editors, *Heads in Grammatical Theory*, pages 292–315. Cambridge University Press.

Parser Evaluation over Local and Non-Local Deep Dependencies in a Large Corpus

Emily M. Bender[♠], **Dan Flickinger**[♡], **Stephan Oepen**[♣], **Yi Zhang**[♢]
[♠]Dept of Linguistics, University of Washington, [♡]CSLI, Stanford University
[♣]Dept of Informatics, Universitetet i Oslo, [♢]Dept of Computational Linguistics, Saarland University
ebender@uw.edu, danf@stanford.edu, oe@ifi.uio.no, yzhang@coli.uni-sb.de

Abstract

In order to obtain a fine-grained evaluation of parser accuracy over naturally occurring text, we study 100 examples each of ten reasonably frequent linguistic phenomena, randomly selected from a parsed version of the English Wikipedia. We construct a corresponding set of gold-standard target dependencies for these 1000 sentences, operationalize mappings to these targets from seven state-of-the-art parsers, and evaluate the parsers against this data to measure their level of success in identifying these dependencies.

1 Introduction

The terms "deep" and "shallow" are frequently used to characterize or contrast different approaches to parsing. Inevitably, such informal notions lack a clear definition, and there is little evidence of community consensus on the relevant dimension(s) of depth, let alone agreement on applicable metrics. At its core, the implied dichotomy of approaches alludes to differences in the interpretation of the parsing task. Its abstract goal, on the one hand, could be pre-processing of the linguistic signal, to enable subsequent stages of analysis. On the other hand, it could be making explicit the (complete) contribution that the grammatical form of the linguistic signal makes to interpretation, working out who did what to whom. Stereotypically, one expects corresponding differences in the choice of interface representations, ranging from various levels of syntactic analysis to logical-form representations of semantics.

In this paper, we seek to probe aspects of variation in automated linguistic analysis. We make the assumption that an integral part of many (albeit not all) applications of parsing technology is the recovery of structural relations, i.e. dependencies at the level of interpretation. We suggest a selection of ten linguistic phenomena that we believe (a) occur with reasonably high frequency in running text and (b) have the potential to shed some light on the depths of linguistic analysis. We quantify the frequency of these constructions in the English Wikipedia, then annotate 100 example sentences for each phenomenon with gold-standard dependencies reflecting core properties of the phenomena of interest. This gold standard is then used to estimate the recall of these dependencies by seven commonly used parsers, providing the basis for a qualitative discussion of the state of the art in parsing for English.

In this work, we answer the call by Rimell et al. (2009) for "construction-focused parser evaluation", extending and complementing their work in several respects: (i) we investigate both local and non-local dependencies which prove to be challenging for many existing state-of-the-art parsers; (ii) we investigate a wider range of linguistic phenomena, each accompanied with an in-depth discussion of relevant properties; and (iii) we draw our data from the 50-million sentence English Wikipedia, which is more varied and a thousand times larger than the venerable WSJ corpus, to explore a more level and ambitious playing field for parser comparison.

2 Background

All parsing systems embody knowledge about possible and probable pairings of strings and corresponding linguistic structure. Such linguistic and probabilistic knowledge can be hand-coded (e.g., as grammar rules) or automatically acquired from labeled or

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 397–408,
Edinburgh, Scotland, UK, July 27–31, 2011.

unlabeled training data. A related dimension of variation is the type of representations manipulated by the parser. We briefly review some representative examples along these dimensions, as these help to position the parsers we subsequently evaluate.[1]

2.1 Approaches to parsing

Source of linguistic knowledge At one end of this dimension, we find systems whose linguistic knowledge is encoded in hand-crafted rules and lexical entries; for English, the ParGram XLE system (Riezler et al., 2002) and DELPH-IN English Resource Grammar (ERG; Flickinger (2000))—each reflecting decades of continuous development—achieve broad coverage of open-domain running text, for example. At the other end of this dimension, we find fully unsupervised approaches (Clark, 2001; Klein and Manning, 2004), where the primary source of linguistic knowledge is co-occurrence patterns of words in unannotated text. As Haghighi and Klein (2006) show, augmenting this knowledge with hand-crafted prototype "seeds" can bring strong improvements. Somewhere between these poles, a broad class of parsers take some or all of their linguistic knowledge from annotated treebanks, e.g. the Penn Treebank (PTB), which encodes "surface grammatical analysis" (Marcus et al., 1993). Such approaches include those that directly (and exclusively) use the information in the treebank (e.g. Charniak (1997), Collins (1999), Petrov et al. (2006), inter alios) as well as those that complement treebank structures with some amount of hand-coded linguistic knowledge (e.g. O'Donovan et al. (2004), Miyao et al. (2004), Hockenmaier and Steedman (2007), inter alios). Another hybrid in terms of its acquisition of linguistic knowledge is the RASP system of Briscoe et al. (2006), combining a hand-coded grammar over PoS tag sequences with a probabilistic tagger and statistical syntactic disambiguation.

Design of representations Approaches to parsing also differ fundamentally in the style of representation assigned to strings. These vary both in their formal nature and the "granularity" of linguistic information (i.e. the number of distinctions assumed), encompassing variants of constituent structure, syntactic dependencies, or logical-form representations of semantics. Parser interface representations range between the relatively simple (e.g. phrase structure trees with a limited vocabulary of node labels as in the PTB, or syntactic dependency structures with a limited vocabulary of relation labels as in Johansson and Nugues (2007)) and the relatively complex, as for example elaborate syntactico-semantic analyses produced by the ParGram or DELPH-IN grammars.

There tends to be a correlation between the methodology used in the acquisition of linguistic knowledge and the complexity of representations: in the creation of a mostly hand-crafted treebank like the PTB, representations have to be simple enough for human annotators to reliably manipulate. Deriving more complex representations typically presupposes further computational support, often involving some hand-crafted linguistic knowledge—which can take the form of mappings from PTB-like representations to "richer" grammatical frameworks (as in the line of work by O'Donovan et al. (2004), and others; see above), or can be rules for creating the parse structures in the first place (i.e. a computational grammar), as for example in the treebanks of van der Beek et al. (2002) or Oepen et al. (2004).[2]

In principle, one might expect that richer representations allow parsers to capture complex syntactic or semantic dependencies more explicitly. At the same time, such "deeper" relations may still be recoverable (to some degree) from comparatively simple parser outputs, as demonstrated for unbounded dependency extraction from strictly local syntactic dependency trees by Nivre et al. (2010).

2.2 An armada of parsers

Stanford Parser (Klein and Manning, 2003) is a probabilistic parser which can produce both phrase structure trees and grammatical relations (syntactic dependencies). The parsing model we evaluate is the

[1] Additional sources of variation among extant parsing technologies include (a) the behavior with respect to ungrammatical inputs and (b) the relationship between probabilistic and symbolic knowledge in the parser, where parsers with a hand-coded grammar at their core typically also incorporate an automatically trained probabilistic disambiguation component.

[2] A noteworthy exception to this correlation is the annotated corpus of Zettlemoyer and Collins (2005), which pairs surface strings from the realm of natural language database interfaces directly with semantic representations in lambda calculus. These were hand-written on the basis of database query statements distributed with the original datasets.

English factored model which combines the preferences of unlexicalized PCFG phrase structures and of lexical dependencies, trained on sections 02–21 of the WSJ portion of the PTB. We chose Stanford Parser from among the state-of-the-art PTB-derived parsers for its support for grammatical relations as an alternate interface representation.

Charniak&Johnson Reranking Parser (Charniak and Johnson, 2005) is a two-stage PCFG parser with a lexicalized generative model for the first-stage, and a discriminative MaxEnt reranker for the second-stage. The models we evaluate are also trained on sections 02–21 of the WSJ. Top-50 readings were used for the reranking stage. The output constituent trees were then converted into Stanford Dependencies. According to Cer et al. (2010), this combination gives the best parsing accuracy in terms of Stanford dependencies on the PTB.

Enju (Miyao et al., 2004) is a probabilistic HPSG parser, combining a hand-crafted core grammar with automatically acquired lexical types from the PTB.[3] The model we evaluate is trained on the same material from the WSJ sections of the PTB, but the treebank is first semi-automatically converted into HPSG derivations, and the annotation is enriched with typed feature structures for each constituent. In addition to HPSG derivation trees, Enju also produces predicate argument structures.

C&C (Clark and Curran, 2007) is a statistical CCG parser. Abstractly similar to the approach of Enju, the grammar and lexicon are automatically induced from CCGBank (Hockenmaier and Steedman, 2007), a largely automatic projection of (the WSJ portion of) PTB trees into the CCG framework. In addition to CCG derivations, the C&C parser can directly output a variant of grammatical relations.

RASP (Briscoe et al., 2006) is an unlexicalized robust parsing system, with a hand-crafted "tag sequence" grammar at its core. The parser thus analyses a lattice of PoS tags, building a parse forest from which the most probable syntactic trees and sets of corresponding grammatical relations can be extracted. Unlike other parsers in our mix, RASP did not build on PTB data in either its PoS tagging or syntactic disambiguation components.

MSTParser (McDonald et al., 2005) is a data-driven dependency parser. The parser uses an edge-factored model and searches for a maximal spanning tree that connects all the words in a sentence into a dependency tree. The model we evaluate is the second-order projective model trained on the same WSJ corpus, where the original PTB phrase structure annotations were first converted into dependencies, as established in the CoNLL shared task 2009 (Johansson and Nugues, 2007).

XLE/ParGram (Riezler et al., 2002, see also Cahill et al., 2008) applies a hand-built Lexical Functional Grammar for English and a stochastic parse selection model. For our evaluation, we used the Nov 4, 2010 release of XLE and the Nov 25, 2009 release of the ParGram English grammar, with c-structure pruning turned off and resource limitations set to the maximum possible to allow for exhaustive search. In particular, we are evaluating the f-structures output by the system.

Each parser, of course, has its own requirements regarding preprocessing of text, especially tokenization. We customized the tokenization to each parser, by using the parser's own internal tokenization or pre-tokenizing to match the parser's desired input. The evaluation script is robust to variations in tokenization across parsers.

3 Phenomena

In this section we summarize the ten phenomena we explore and our motivations for choosing them. Our goal was to find phenomena where the relevant dependencies are relatively subtle, such that more linguistic knowledge is beneficial in order to retrieve them. Though this set is of course only a sampling, these phenomena illustrate the richness of structure, both local and non-local, involved in the mapping from English strings to their meanings. We discuss the phenomena in four sets and then briefly review their representation in the Penn Treebank.

3.1 Long distance dependencies

Three of our phenomena can be classified as involving long-distance dependencies: finite *that*-less relatives clauses ('barerel'), tough adjectives ('tough') and right node raising ('rnr'). These are illustrated

[3] This hand-crafted grammar is distinct from the ERG, despite sharing the general framework of HPSG. The ERG is not included in our evaluation, since it was used in the extraction of the original examples and thus cannot be fairly evaluated.

in the following examples:[4]

(1) barerel: This is the second *time* in a row Australia **has lost** their home tri-nations' series.

(2) tough: Original *copies* are very hard to **find**.

(3) rnr: Ilúvatar, as his names imply, exists **before** and **independently of** *all else*.

While the majority of our phenomena involve local dependencies, we include these long-distance dependency types because they are challenging for parsers and enable more direct comparison with the work of Rimell et al. (2009), who also address right node raising and bare relatives. Our barerel category corresponds to their "object reduced relative" category with the difference that we also include adverb relatives, where the head noun functions as a modifier within the relative clause, as does *time* in (1). In contrast, our rnr category is somewhat narrower than Rimell et al. (2009)'s "right node raising" category: where they include raised modifiers, we restrict our category to raised complements.

Part of the difficulty in retrieving long-distance dependencies is that the so-called extraction site is not overtly marked in the string. In addition to this baseline level of complication, these three construction types present further difficulties: Bare relatives, unlike other relative clauses, do not carry any lexical cues to their presence (i.e., no relative pronouns). *Tough* adjective constructions require the presence of specific lexical items which form a subset of a larger open class. They are rendered more difficult by two sources of ambiguity: alternative subcategorization frames for the adjectives and the purposive adjunct analysis (akin to *in order to*) for the infinitival VP. Finally, right node raising often involves coordination where one of the conjuncts is in fact not a well-formed phrase (e.g., *independently of* in (3)), making it potentially difficult to construct the correct coordination structure, let alone associate the raised element with the correct position in each conjunct.

3.2 Non-dependencies

Two of our phenomena crucially look for the lack of dependencies. These are *it* expletives ('itexpl') and verb-particle constructions ('vpart'):

(4) itexpl: Crew negligence is blamed, and *it* **is suggested** that the flight crew were drunk.

(5) vpart: He once **threw out** two *baserunners* at home in the same inning.

The English pronoun *it* can be used as an ordinary personal pronoun or as an expletive: a placeholder for when the language demands a subject (or occasionally object) NP but there is no semantic role for that NP. The expletive *it* only appears when it is licensed by a specific construction (such as extraposition, (4)) or selecting head. If the goal of parsing is to recover from the surface string the dependencies capturing who did what to whom, expletive *it* should not feature in any of those dependencies. Likewise, instances of expletive *it* should be detected and discarded in reference resolution. We hypothesize that detecting expletive *it* requires encoding linguistic knowledge about its licensers.

The other non-dependency we explore is between the particle in verb-particle constructions and the direct object. Since English particles are almost always homophonous with prepositions, when the object of the verb-particle pair follows the particle, there will always be a competing analysis which analyses the sequence as V+PP rather than V+particle+NP. Furthermore, since verb-particle pairs often have non-compositional semantics (Sag et al., 2002), misanalyzing these constructions could be costly to downstream components.

3.3 Phrasal modifiers

Our next category concerns modifier phrases:

(6) ned: *Light* ***colored*** **glazes** also have softening effects when painted over dark or bright images.

(7) absol: The format **consisted** of 12 games, each *team* ***facing*** the other teams twice.

The first, ('ned'), is a pattern which to our knowledge has not been named in the literature, where a noun takes the typically verbal *-ed* ending, is modified by another noun or adjective, and functions as a modifier or a predicate. We believe this phenomenon to be interesting because its unusual morphology is likely to lead PoS-taggers astray, and because the often-hyphenated Adj+N-ed constituent has productive internal structure constraining its interpretation.

The second phrasal modifier we investigate is the absolutive construction. An absolutive consists of an

[4] All examples are from our data. Words involved in the relevant dependencies are highlighted in italics (dependents) and boldface (heads).

NP followed by a non-finite predicate (such as could appear after the copula *be*). The whole phrase modifies a verbal projection that it attaches to. Absolutives may be marked with *with* or unmarked. Here, we focus on the unmarked type as this lack of lexical cue can make the construction harder to detect.

3.4 Subtle arguments

Our final three phenomena involve ways in which verbal arguments can be more difficult to identify than in ordinary finite clauses. These include detecting the arguments of verbal gerunds ('vger'), the interleaving of arguments and adjuncts ('argadj') and raising/control ('control') constructions.

(8) vger: ***Accessing*** *the website* without the "www" subdomain **returned** a copy of the main site for "EP.net".

(9) argadj: The story **shows**, *through* flashbacks, the different *histories* of the characters.

(10) control: *Alfred* "retired" in 1957 at age 60 but **continued** to **paint** full time.

In a verbal gerund, the *-ing* form a verb retains verbal properties (e.g., being able to take NP complements, rather than only PP complements) but heads a phrase that fills an NP position in the syntax (Malouf, 2000). Since gerunds have the same morphology as present participle VPs, their role in the larger clause is susceptible to misanalysis.

The argadj examples are of interest because English typically prefers to have direct objects directly adjacent to the selecting verb. Nonetheless, phenomena such as parentheticals and heavy-NP shift (Arnold et al., 2000), in which "heavy" constituents appear further to the right in the string, allow for adjunct-argument order in a minority of cases. We hypothesize that the relative infrequency of this construction will lead parsers to prefer incorrect analyses (wherein the adjunct is picked up as a complement, the complement as an adjunct or the structure differs entirely) unless they have access to linguistic knowledge providing constraints on possible and probable complementation patterns for the head.

Finally, we turn to raising and control verbs ('control') (e.g., Huddleston and Pullum (2002, ch. 14)). These verbs select for an infinitival VP complement and stipulate that another of their arguments (subject or direct object in the examples we explore) is identified with the unrealized subject position of the infinitival VP. Here it is the dependency between the infinitval VP and the NP argument of the "upstairs" verb which we expect to be particularly subtle. Getting this right requires specific lexical knowledge about which verbs take these complementation patterns. This lexical knowledge needs to be represented in such a way that it can be used robustly even in the case of passives, relative clauses, etc.[5]

3.5 Penn Treebank representations

We investigated the representation of these 10 phenomena in the PTB (Marcus et al., 1993) in two steps: First we explored the PTB's annotation guidelines (Bies et al., 1995) to determine how the relevant dependencies were intended to be represented. We then used Ghodke and Bird's (2010) Treebank Search to find examples of the intended annotations as well as potential examples of the phenomena annotated differently, to get a sense of the consistency of the annotation from both precision and recall perspectives. In this study, we take the phrase structure trees of the PTB to represent dependencies based on reasonable identification of heads.

The barerel, vpart, and absol phenomena are completely unproblematic, with their relevant dependencies explicitly and reliably represented. In addition, the tough construction is reliably annotated, though one of the dependencies we take to be central is not directly represented: The missing argument is linked to a null *wh* head at the left edge of the complement of the tough predicate, rather than to its subject. Two further phenomena (rnr and vger) are essentially correctly represented: the representations of the dependencies are explicit and mostly but not entirely consistently applied. Two out of a sample of 20 examples annotated as containing rnr did not, and two out of a sample of 35 non-rnr-annotated coordinations actually contained rnr. For vger the primary problem is with the PoS tagging, where the gerund is sometimes given a nominal tag, contrary to PTB guidelines, though the structure above it conforms.

The remaining four constructions are more problematic. In the case of object control, while the guide-

[5] Distinguishing between raising and control requires further lexical knowledge and is another example of a "non-dependency" (in the raising examples). We do not draw that distinction in our annotations.

lines specify an analysis in which the shared NP is attached as the object of the higher verb, the PTB includes not only structures conforming to that analysis but also "small clause" structures, with the latter obscuring the relationship of the shared argument to the higher verb. In the case of itexpl, the adjoined `(S(-NONE- *EXP*))` indicating an expletive use of *it* is applied consistently for extraposition (as prescribed in the guidelines). However, the set of lexical licensers of the expletive is incomplete. For argadj we run into the problem that the PTB does not explicitly distinguish between post-verbal modifiers and verbal complements in the way that they are attached. The guidelines suggest that the function tags (e.g., `PP-LOC`, etc.) should allow one to distinguish these, but examination of the PTB itself suggests that they are not consistently applied. Finally, the ned construction is not mentioned in the PTB guidelines nor is its internal structure represented in the treebank. Rather, strings like *gritty-eyed* are left unsegmented and tagged as `JJ`.

We note that the PTB representations of many of these phenomena (barerel, tough, rnr, argadj, control, itexpl) involve empty elements and/or function tags. Systems that strip these out before training, as is common practice, will not benefit from the information that is in the PTB.

Our purpose here is not to criticize the PTB, which has been a tremendously important resource to the field. Rather, we have two aims: The first is to provide context for the evaluation of PTB-derived parsers on these phenomena. The second is to highlight the difficulty of producing consistent annotations of any complexity as well as the hurdles faced by a hand-annotation approach when attempting to scale a resource to more complex representations and/or additional phenomena (though cf. Vadas and Curran (2008) on improving PTB representations).

4 Methodology

4.1 Data extraction

We processed 900 million tokens of Wikipedia text using the October 2010 release of the ERG, following the work of the WikiWoods project (Flickinger et al., 2010). Using the top-ranked ERG derivation trees as annotations over this corpus and simple patterns using names of ERG-specific constructions or lexical types, we randomly selected a set of candidate sentences for each of our ten phenomena. These candidates were then hand-vetted in sequence by two annotators to identify, for each phenomenon, 100 examples that do in fact involve the phenomenon in question and which are both grammatical and free of typos. Examples that were either deemed overly basic (e.g. plain V + V coordination, which the ERG treats as rnr) or inappropriately complex (e.g. non-constituent coordination obscuring the interleaving of arguments and adjuncts) were also discarded at this step. Table 1 summarizes relative frequencies of each phenomenon in about 47 million parsed Wikipedia sentences, as well as the total size of the candidate sets inspected. For the control and tough phenomena hardly any filtering for complexity was applied, hence these can serve as indicators of the rate of genuine false positives. For phenomena that partially overlap with those of Rimell et al. (2009), it appears our frequency estimates are comparable to what they report for the Brown Corpus (but not the WSJ portion of the PTB).

Phenomenon	Frequency	Candidates
barerel	2.12%	546
tough	0.07%	175
rnr	0.69%	1263
itexpl	0.13%	402
vpart	4.07%	765
ned	1.18%	349
absol	0.51%	963
vger	5.16%	679
argadj	3.60%	1346
control	3.78%	124

Table 1: Relative frequencies of phenomena matches in Wikipedia, and number of candidate strings vetted.

4.2 Annotation format

We annotated up to two dependency triples per phenomenon instance, identifying the heads and dependents by the surface form of the head words in the sentence suffixed with a number indicating word position (see Table 2).[6] Some strings contain more than one instance of the phenomenon they illustrate; in these cases, multiple sets of dependencies are

[6] As the parsers differ in tokenization strategies, our evaluation script treats these position IDs as approximate indicators.

Item ID	Phenomenon	Polarity	Dependency
1011079100200	absol	1	having-2|been-3|passed-4 ARG act-1
1011079100200	absol	1	withdrew-9 MOD having-2|been-3|passed-4
1011079100200	absol	1	carried+on-12 MOD having-2|been-3|passed-4

Table 2: Sample annotations for sentence # 1011079100200: *The-0 act-1 having-2 been-3 passed-4 in-5 that-6 year-7 Jessop-8 withdrew-9 and-10 Whitworth-11 carried-12 on-13 with-14 the-15 assistance-16 of-17 his-18 son-19.*

Phenomenon	Head	Type	Dependent	Distance
Bare relatives	gapped predicate in relative	ARG2/MOD	modified noun	3.0 (8)
(barerel)	modified noun	MOD	top predicate of relative	3.3 (8)
Tough adjectives	*tough* adjective	ARG2	*to*-VP complement	1.7 (5)
(tough)	gapped predicate in *to*-VP	ARG2	subject/modifiee of adjective	6.4 (21)
Right Node Raising	verb/prep2	ARG2	shared noun	2.8 (9)
(rnr)	verb/prep1	ARG2	shared noun	6.1 (12)
Expletive It	*it*-subject taking verb	!ARG1	*it*	1.2 (3)
(itexpl)	raising-to-object verb	!ARG2	*it*	–
Verb+particle constructions	particle	!ARG2	complement	2.7 (9)
(vpart)	verb+particle	ARG2	complement	3.7 (10)
Adj/Noun2 + Noun1-*ed*	head noun	MOD	Noun1-*ed*	2.4 (17)
(ned)	Noun1-*ed*	ARG1/MOD	Adj/Noun2	1.0 (1.5)
Absolutives	absolutive predicate	ARG1	subject of absolutive	1.7 (12)
(absol)	main clause predicate	MOD	absolutive predicate	9.8 (26)
Verbal gerunds	selecting head	ARG[1,2]	gerund	1.9 (13)
(vger)	gerund	ARG2/MOD	first complement/modifier of gerund	2.3 (8)
Interleaved arg/adj	selecting verb	MOD	interleaved adjunct	1.2 (7)
(argadj)	selecting verb	ARG[2,3]	displaced complement	5.9 (26)
Control	"upstairs" verb	ARG[2,3]	"downstairs" verb	2.4 (23)
(control)	"downstairs" verb	ARG1	shared argument	4.8 (17)

Table 3: Dependencies labeled for each phenomenon type, including average and maximum surface distances.

recorded. In addition, some strings evince more than one of the phenomena we are studying. However, we only annotate the dependencies associated with the phenomenon the string was selected to represent. Finally, in examples with coordinated heads or dependents, we recorded separate dependencies for each conjunct. In total, we annotated 2127 dependency triples for the 1000 sentences, including 253 negative dependencies (see below). Table 3 outlines the dependencies annotated for each phenomenon.

To allow for multiple plausible attachment sites, we give disjunctive values for heads or dependents in several cases: (i) with auxiliaries, (ii) with complementizers (*that* or *to*, as in Table 2), (iii) in cases of measure or classifier nouns or partitives, (iv) with multi-word proper names and (v) where there is genuine attachment ambiguity for modifiers. As these sets of targets are disjunctive, these conventions should have the effect of increasing measured parser performance. 580 (27%) of the annotated dependencies had at least one disjunction.

4.3 Annotation and reconciliation process

The entire data set was annotated independently by two annotators. Both annotators were familiar with the ERG, used to identify these sentences in the WikiWoods corpus, but the annotation was done without reference to the ERG parses. Before beginning annotation on each phenomenon, we agreed on which dependencies to annotate. We also communicated with each other about annotation conventions as the need for each convention became clear. The annotation conventions address how to handle coordination, semantically empty auxiliaries, passives and similar orthogonal phenomena.

Once the entire data set was dual-annotated, we compared annotations, identifying the following sources of mismatch: typographical errors, incompletely specified annotation conventions, inconsistent application of conventions (101 items, dropping in frequency as the annotation proceeded), and genuine disagreement about what to annotate, either different numbers of dependencies of interest identified

in an item (59 items) or conflicting elements in a dependency (54 items).[7] Overall, our initial annotation pass led to agreement on 79% of the items, and a higher per-dependency level of agreement. Agreement could be expected to approach 90% with more experience in applying annotation conventions.

We then reconciled the annotations, using the comparison to address all sources of difference. In most cases, we readily agreed which annotation was correct and which was in error. In a few cases, we decided that both annotations were plausible alternatives (e.g., in terms of alternative attachment sites for modifiers) and so created a single merged annotation expressing the disjunction of both (cf. § 4.2).

5 Evaluation

With the test data consisting of 100 items for each of our ten selected phenomena, we ran all seven parsing systems and recorded their dependency-style outputs for each sentence. While these outputs are not directly comparable with each other, we were able to associate our manually-annotated target dependencies with parser-specific dependencies, by defining sets of phenomenon-specific regular expressions for each parser. In principle, we allow this mapping to be somewhat complex (and forgiving to non-contentful variation), though we require that it work deterministically and not involve specific lexical information. An example set is given in Fig. 2.

```
"absol" =>
{'ARG1' => [
'\(ncsubj \W*{W1}\W*_(\d+) \W*{W2}\W*_(\d+) _\)',
'\(ncmod _ \W*{W2}\W*_(\d+) \W*{W1}\W*_(\d+)\)'],
'ARG' => [
'\(ncsubj \W*{W1}\W*_(\d+) \W*{W2}\W*_(\d+) _\)',
'\(ncmod _ \W*{W1}\W*_(\d+) \W*{W2}\W*_(\d+)\)'],
'MOD' => [
'\(xmod _ \W*{W1}\W*_(\d+) \W*{W2}\W*_(\d+)\)',
'\(ncmod _ \W*{W1}\W*_(\d+) \W*{W2}\W*_(\d+)\)',
'\(cmod _ \W*{W1}\W*_(\d+) \W*{W2}\W*_(\d+)\)']}
```

Figure 2: Regexp set to evaluate C&C for absol.

These expressions fit the output that we got from the C&C parser, illustrated in Fig. 3 with a relevant portion of the dependencies produced for the example in Table 2. Here the C&C dependency `(ncsubj passed_4 Act_1 _)` matches the first target in the gold-standard (Table 2), but no matching C&C dependency is found for the other two targets.

```
(xmod _ Act_1 passed_4)
(ncsubj passed_4 Act_1 _)
(ncmod _ withdrew,_9 Jessop_8)
(dobj year,_7 withdrew,_9)
```

Figure 3: Excerpts of C&C output for item in Table 2.

The regular expressions operate solely on the dependency labels and are not lexically-specific. They are specific to each phenomenon, as we did not attempt to write a general dependency converter, but rather to discover what patterns of dependency relations describe the phenomenon when it is correctly identified by each parser. Thus, though we did not hold out a test set, we believe that they would generalize to additional gold standard material annotated in the same way for the same phenomena.[8]

In total, we wrote 364 regular expressions to handle the output of the seven parsers, allowing some leeway in the role labels used by a parser for any given target dependency. The supplementary materials for this paper include the test data, parser outputs, target annotations, and evaluation script.

Fig. 1 provides a visualization of the results of our evaluation. Each column of points represents one dependency type. Dependency types for the same phenomenon are represented by adjacent columns. The order of the columns within a phenomenon follows the order of the dependency descriptions in Table 3: For each pair, the dependency type with the higher score for the majority of the parsers is shown first (to the left). The phenomena themselves are also arranged according to increasing (average) difficulty. itexpl only has one column, as we annotated just one dependency per instance here. (The two descriptions in Table 3 reflect different, mutually-incompatible instance types.) Since expletive *it* should not be the semantic dependent of any head, the targets are generalized for this phenomenon and the evaluation script counts as incor-

[7] We do not count typographical errors or incompletely specified conventions as failures of inter-annotator agreement.

[8] In the case of the XLE, our simplistic regular-expression approach to the interpretation of parser outputs calls for much more complex patterns than for the other parsers. This is owed to the rich internal structure of LFG f-structures and higher granularity of linguistic analysis, where feature annotations on nodes as well as reentrancies need to be taken into account. Therefore, our current results for the XLE admit small amounts of both over- and under-counting.

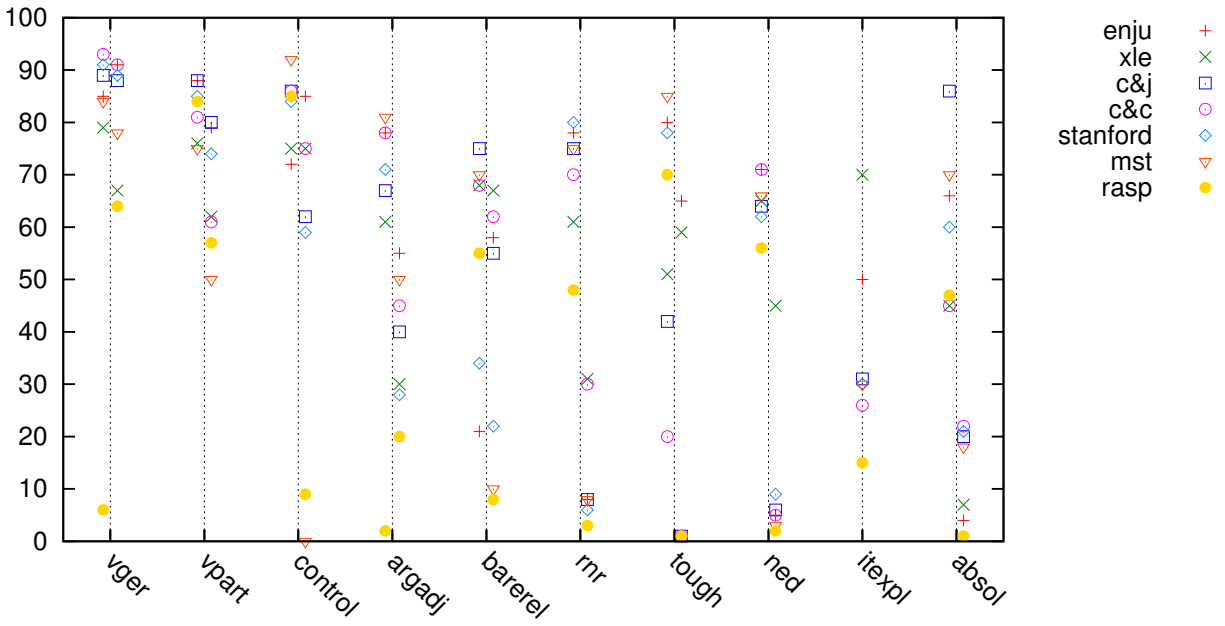


Figure 1: Individual dependency recall for seven parsers over ten phenomena.

rect any dependency involving referential *it*.

We observe fairly high recall of the dependencies for vpart and vger (with the exception of RASP), and high recall for both dependencies representing control for five systems. While Enju, Stanford, MST, and RASP all found between 70 and 85% of the dependency between the adjective and its complement in the tough construction, only Enju and XLE represented the dependency between the subject of the adjective and the gap inside the adjective's complement. For the remaining phenomena, each parser performed markedly worse on one dependency type, compared to the other. The only exceptions here are XLE and C&C's (and to a lesser extent, C&J's) scores for barerel. No system scored higher than 33% on the harder of the two dependencies in rnror absol, and Stanford, MST, and RASP all scored below 25% on the harder dependency in barerel. Only XLE scored higher than 10% on the second dependency for ned and higher than 50% for itexpl.

6 Discussion

From the results in Fig. 1, it is clear that even the best of these parsers fail to correctly identify a large number of relevant dependencies associated with linguistic phenomena that occur with reasonable frequency in the Wikipedia. Each of the parsers attempts with some success to analyze each of these phenomena, reinforcing the claim of relevance, but they vary widely across phenomena. For the two long-distance phenomena that overlap with those studied in Rimell et al. (2009), our results are comparable.[9] Our evaluation over Wikipedia examples thus shows the same relative lack of success in recovering long-distance dependencies that they found for WSJ sentences. The systems did better on relatively well-studied phenomena including control, vger, and vpart, but had less success with the rest, even though all but two of those remaining phenomena involve syntactically local dependencies (as indicated in Table 3).

Successful identification of the dependencies in these phenomena would, we hypothesize, benefit from richer (or deeper) linguistic information when parsing, whether it is lexical (tough, control, itexpl, and vpart), or structural (rnr, absol, vger, argadj, and barerel), or somewhere in between, as with ned. In the case of treebank-trained parsers, for the information to be available, it must be consistently encoded in the treebank and attended to during training. As

[9] Other than Enju, which scores 16 points higher in the evaluation of Rimell et al., our average scores for each parser across the dependencies for these phenomena are within 12 points of those reported by Rimell et al. (2009) and Nivre et al. (2010).

noted in Sections 2.1 and 3.5, there is tension between developing sufficiently complex representations to capture linguistic phenomena and keeping an annotation scheme simple enough that it can be reliably produced by humans, in the case of hand-annotation.

7 Related Work

This paper builds on a growing body of work which goes beyond (un)labeled bracketing in parser evaluation, including Lin (1995), Carroll et al. (1998), Kaplan et al. (2004), Rimell et al. (2009), and Nivre et al. (2010). Most closely related are the latter two of the above, as we adopt their "construction-focused parser evaluation methodology".

There are several methodological differences between our work and that of Rimell et al. First, we draw our evaluation data from a much larger and more varied corpus. Second, we automate the comparison of parser output to the gold standard, and we distribute the evaluation scripts along with the annotated corpus, enhancing replicability. Third, where Rimell et al. extract evaluation targets on the basis of PTB annotations, we make use of a linguistically precise broad-coverage grammar to identify candidate examples. This allows us to include both local and non-local dependencies not represented or not reliably encoded in the PTB, enabling us to evaluate parser performance with more precision over a wider range of linguistic phenomena.

These methodological innovations bring two empirical results. The first is qualitative: Where previous work showed that overall Parseval numbers hide difficulties with long-distance dependencies, our results show that there are multiple kinds of reasonably frequent *local* dependencies which are also difficult for the current standard approaches to parsing. The second is quantitative: Where Rimell et al. found two phenomena which were virtually unanalyzed (recall below 10%) for one or two parsers each, we found eight phenomena which were virtually unanalyzed by at least one system, including two unanalyzed by five and one by six. Every system had at least one virtually unanalyzed phenomenon. Thus we have shown that the dependencies being missed by typical modern approaches to parsing are more varied and more numerous than previously thought.

8 Conclusion

We have presented a detailed construction-focused evaluation of seven parsers over 10 phenomena, with 1000 examples drawn from English Wikipedia. Gauging recall of such "deep" dependencies, in our view, can serve as a proxy for downstream processing involving semantic interpretation of parser outputs. Our annotations and automated evaluation script are provided in the supplementary materials, for full replicability. Our results demonstrate that significant opportunities remain for parser improvement, and highlight specific challenges that remain invisible in aggregate parser evaluation (e.g. Parseval or overall dependency accuracy). These results suggest that further progress will depend on training data that is both more extensive and more richly annotated than what is typically used today (seeing, for example, that a large part of more detailed PTB annotation remains ignored in much parsing work).

There are obvious reasons calling for diversity in approaches to parsing and for different trade-offs in, for example, the granularity of linguistic analysis, average accuracy, cost of computation, or ease of adaptation. Our proposal is not to substitute construction-focused evaluation on Wikipedia data for widely used aggregate metrics and reference corpora, but rather to augment such best practices in the spirit of Rimell et al. (2009) and expand the range of phenomena considered in such evaluations. Across frameworks and traditions (and in principle languages), it is of vital importance to be able to evaluate the quality of parsing (and grammar induction) algorithms in a maximally informative manner.

Acknowledgments

We are grateful to Tracy King for her assistance in setting up the XLE system and to three anonymous reviewers for helpful comments. The fourth author thanks DFKI and the DFG funded Excellence Cluster on MMCI for their support of the work. Data preparation on the scale of Wikipedia was made possible through access to large-scale HPC facilities, and we are grateful to the Scientific Computing staff at UiO and the Norwegian Metacenter for Computational Science.

References

Jennifer E. Arnold, Thomas Wasow, Anthony Losongco, and Ryan Ginstrom. 2000. Heaviness vs. newness: The effects of structural complexity and discourse status on constituent ordering. *Language*, 76(1):28–55.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for treebank II style Penn treebank project. Technical report, University of Pennsylvania.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sydney, Australia.

Aoife Cahill, John T. Maxwell III, Paul Meurer, Christian Rohrer, and Victoria Rosén. 2008. Speeding up LFG parsing using c-structure pruning. In *Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, pages 33–40, Manchester, England, August. Coling 2008 Organizing Committee.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: A survey and a new proposal. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 447–454, Granada.

Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to Stanford dependencies: Trade-offs between speed and accuracy. In *7th International Conference on Language Resources and Evaluation (LREC 2010)*, Malta.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598 – 603, Providence, RI.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 5th Conference on Natural Language Learning*, pages 105–112. Toulouse, France.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestøl. 2010. WikiWoods. Syntacto-semantic annotation for English Wikipedia. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Valletta, Malta.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):15 – 28.

Sumukh Ghodke and Steven Bird. 2010. Fast query for large treebanks. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 267–275, Los Angeles, California, June. Association for Computational Linguistics.

Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 881–888, Sydney, Australia, July. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Rodney Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, Cambridge, UK.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *In Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia.

Ron Kaplan, Stefan Riezler, Tracy H King, John T Maxwell III, Alex Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In Susan Dumais, Daniel Marcu, and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 97–104, Boston, Massachusetts, USA, May. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15*, pages 3–10, Cambridge, MA. MIT Press.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 478–485, Barcelona, Spain.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*, pages 1420–1425, Montreal, Canada.

Robert Malouf. 2000. Verbal gerunds as mixed categories in HPSG. In Robert Borsley, editor, *The Nature*

and Function of Syntactic Categories, pages 133–166. Academic Press, New York.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, Canada.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proceedings of the 1st International Joint Conference on Natural Language Processing*, pages 684–693, Hainan Island, China.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 833–841, Beijing, China.

Ruth O'Donovan, Michael Burke, Aoife Cahill, Josef Van Genabith, and Andy Way. 2004. Large-scale induction and evaluation of lexical resources from the penn-ii treebank. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 367–374, Barcelona, Spain.

Stephan Oepen, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation*, 2(4):575 – 596.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, Philadelphia, PA.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821, Singapore. Association for Computational Linguistics.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions. A pain in the neck for NLP. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 189–206. Springer, Berlin, Germany.

David Vadas and James R. Curran. 2008. Parsing noun phrase structure with CCG. In *Proceedings of ACL-08: HLT*, pages 335–343, Columbus, Ohio, June. Association for Computational Linguistics.

L. van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino Dependency Treebank. In Mariet Theune, Anton Nijholt, and Hendri Hondorp, editors, *Computational Linguistics in the Netherlands*, Amsterdam, The Netherlands. Rodopi.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Annual Conference on Uncertainty in Artificial Intelligence*, pages 658–666, Arlington, Virginia. AUAI Press.

Learning to Simplify Sentences with Quasi-Synchronous Grammar and Integer Programming

Kristian Woodsend and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
k.woodsend@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Text simplification aims to rewrite text into simpler versions, and thus make information accessible to a broader audience. Most previous work simplifies sentences using handcrafted rules aimed at splitting long sentences, or substitutes difficult words using a predefined dictionary. This paper presents a data-driven model based on quasi-synchronous grammar, a formalism that can naturally capture structural mismatches and complex rewrite operations. We describe how such a grammar can be induced from Wikipedia and propose an integer linear programming model for selecting the most appropriate simplification from the space of possible rewrites generated by the grammar. We show experimentally that our method creates simplifications that significantly reduce the reading difficulty of the input, while maintaining grammaticality and preserving its meaning.

1 Introduction

Sentence simplification is perhaps one of the oldest text rewriting problems. Given a *source* sentence, the goal is to create a grammatical *target* that is easier to read with simpler vocabulary and syntactic structure. An example is shown in Table 1 involving a broad spectrum of rewrite operations such as deletion, substitution, insertion, and reordering. The popularity of the simplification task stems from its potential relevance to various applications. Examples include the development of reading aids for people with aphasia (Carroll et al., 1999), non-native speakers (Siddharthan, 2003) and more generally individuals with low literacy (Watanabe et al., 2009). A simplification component could be also used as a preprocessing step to improve the performance of parsers (Chandrasekar et al., 1996), summarizers (Beigman Klebanov et al., 2004) and semantic role labelers (Vickrey and Koller, 2008).

Also contributing to the firmness in copper, the analyst noted, was a report by Chicago purchasing agents, which precedes the full purchasing agents report that is due out today and gives an indication of what the full report might hold.
Also contributing to the firmness in copper, the analyst noted, was a report by Chicago purchasing agents. The Chicago report precedes the full purchasing agents report. The Chicago report gives an indication of what the full report might hold. The full report is due out today.

Table 1: Example of a source sentence (top) and its simplification (bottom).

Simplification is related to, but different from paraphrase extraction (Barzilay, 2003). We must not only have access to paraphrases (i.e., rewrite rules), but also be able to combine them to generate new text, in a simpler language. The task is also distinct from sentence compression as it aims to render a sentence more accessible while preserving its meaning. On the contrary, compression unavoidably leads to some information loss as it creates shorter sentences without necessarily reducing complexity. In fact, one of the commonest simplification operations is sentence splitting which usually produces longer rather than shorter output! Moreover, mod-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 409–420,
Edinburgh, Scotland, UK, July 27–31, 2011.

els developed for sentence compression have been mostly designed with one rewrite operation in mind, namely word deletion, and are thus unable to model consistent syntactic effects such as reordering, sentence splitting, changes in non-terminal categories, and lexical substitution (but see Cohn and Lapata 2008 and Zhao et al. 2009 for notable exceptions).

In this paper we propose a sentence simplification model that is able to handle structural mismatches and complex rewriting operations. Our approach is based on quasi-synchronous grammar (QG, Smith and Eisner 2006), a formalism that is well suited for text rewriting. Rather than postulating a strictly synchronous structure over the source and target sentences, QG identifies a "sloppy" alignment of parse trees assuming that the target tree is in some way "inspired by" the source tree. Specifically, our model is formulated as an integer linear program and uses QG to capture the space of all possible rewrites. Given a source tree, it finds the best target tree licensed by the grammar subject to constraints such as sentence length and reading ease. Our model is conceptually simple and computationally efficient. Furthermore, it finds globally optimal simplifications without resorting to heuristics or approximations during the decoding process.

Contrary to most previous approaches (see the discussion in Section 2) which rely heavily on hand-crafted rules, our model *learns* simplification rewrites automatically from examples of source-target sentences. Our work joins others in using Wikipedia to extract data appropriate for model training (Yamangil and Nelken, 2008; Yatskar et al., 2010; Zhu et al., 2010). Advantageously, the Simple English Wikipedia (henceforth SimpleEW) provides a large repository of simplified language; it uses fewer words and simpler grammar than the ordinary English Wikipedia (henceforth MainEW) and is aimed at non-native English speakers, children, translators, people with learning disabilities or low reading proficiency. We exploit Wikipedia and create a (parallel) simplification corpus in two ways: by aligning MainEW sentences to their SimpleEW counterparts, and by extracting training instances from SimpleEW revision histories, thus leveraging Wikipedia's collaborative editing process.

Our experimental results demonstrate that a simplification model can be learned from Wikipedia data alone without any manual effort. Perhaps unsurprisingly, the quality of the QG grammar rules greatly improves when these are learned from revision histories which are less noisy than sentence alignments. When compared against current state-of-the-art methods (Zhu et al., 2010) our model yields significantly simpler output that is both grammatical and meaning preserving.

2 Related Work

Sentence simplification has attracted a great deal of attention due to its potential impact on society. The literature is rife with attempts to simplify text using mostly hand-crafted syntactic rules aimed at splitting long and complicated sentences into several simpler ones (Carroll et al., 1999; Chandrasekar et al., 1996; Siddharthan, 2004; Vickrey and Koller, 2008). Other work focuses on lexical simplifications and substitutes difficult words by more common WordNet synonyms or paraphrases found in a predefined dictionary (Devlin, 1999; Inui et al., 2003; Kaji et al., 2002).

More recently, Yatskar et al. (2010) explore data-driven methods to learn lexical simplifications from Wikipedia revision histories. A key idea in their work is to utilize SimpleEW edits, while recognizing that these may serve other functions, such as vandalism removal or introduction of new content. Zhu et al. (2010) also use Wikipedia to learn a sentence simplification model which is able to perform four rewrite operations, namely substitution, reordering, splitting, and deletion. Inspired by syntax-based SMT (Yamada and Knight, 2001), their model consists of three components: a language model $P(\mathbf{s})$ whose role is to guarantee that the simplification output is grammatical, a direct translation model $P(\mathbf{s}|\mathbf{c})$ capturing the probability that the target sentence $\mathbf{s}$ is a simpler version of the source $\mathbf{c}$, and a decoder which searches for the simplification $\mathbf{s}$ which maximizes $P(\mathbf{s})P(\mathbf{s}|\mathbf{c})$. The translation model is the product of the aforementioned four rewrite operations whose probabilities are estimated from a parallel corpus of MainEW and SimpleEW sentences using an expectation maximization algorithm. Their decoder translates sentences into simpler alternatives by greedily selecting the branch in the source tree with the highest probability.

Our own work formulates sentence simplification in the framework of Quasi-synchronous grammar (QG, Smith and Eisner 2006). QG allows to describe non-isomorphic tree pairs (the grammar rules can comprise trees of arbitrary depth, and fragments can be mapped) and is thus suited to text-rewriting tasks which typically involve a number of local modifications to the input text. We use quasi-synchronous grammar to learn a wide range of rewrite operations capturing both lexical and structural simplifications naturally without any additional rule engineering. In contrast to Yatskar et al. (2010) and Zhu et al. (2010), simplification operations (e.g., substitution or splitting) are not modeled explicitly; instead, we leave it up to our grammar extraction algorithm to learn appropriate rules that reflect the training data. Compared to Zhu et al., our model is conceptually simpler and more general. The proposed ILP formulation not only allows to efficiently search through the space of many QG rules but also to incorporate constraints relating to grammaticality and the task at hand without the added computational cost of integrating a language model. Furthermore, our learning framework is not limited to simplification and could be easily adapted to other rewriting tasks. Indeed, the QG formalism has been previously applied to parser adaptation and projection (Smith and Eisner, 2009), paraphrase identification (Das and Smith, 2009), question answering (Wang et al., 2007), and title generation (Woodsend et al., 2010).

Finally, our work relates to a large body of recent literature on Wikipedia and its potential for a wide range of NLP tasks. Beyond text rewriting, examples include semantic relatedness (Ponzetto and Strube, 2007), information extraction (Wu and Weld, 2010), ontology induction (Nastase and Strube, 2008), and the automatic creation of overview articles (Sauper and Barzilay, 2009).

3 Sentence Simplification Model

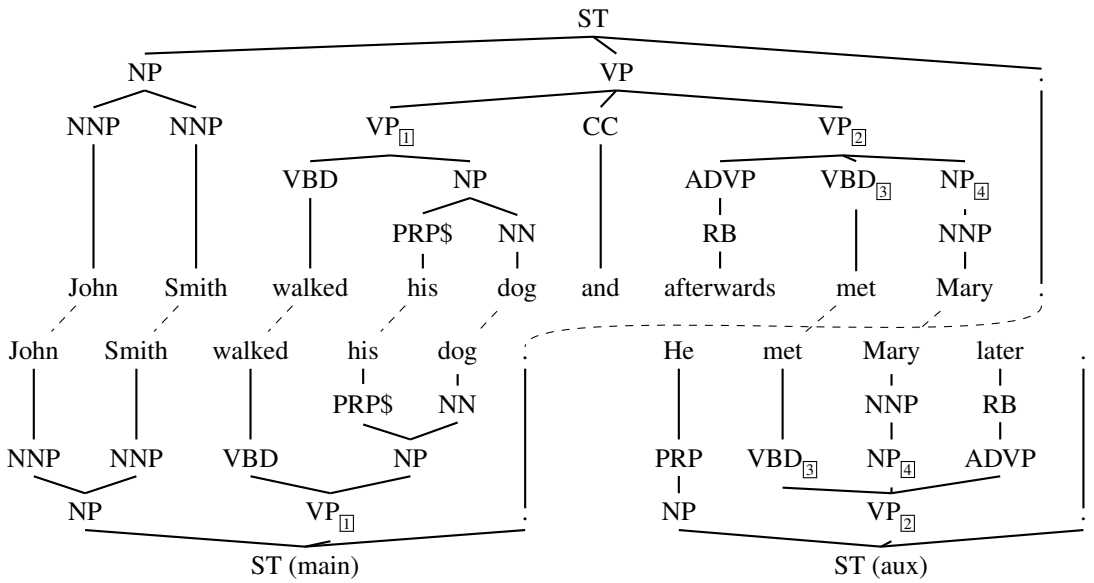
Our model takes a single sentence as input and creates a version that is simpler to read. This may involve rendering syntactically complex structures simpler (e.g., through sentence splitting), or substituting rare words with more common words or phrases (e.g., such that a second language learner may be familiar with), or deleting elements of the original text in order to produce a relatively simpler and shallower syntactic structure. In addition, the output must be grammatical and coherent. These constraints are *global* in their scope, and cannot be adequately satisfied by optimizing each one of them individually. Our approach therefore uses an ILP formulation which will provide a globally optimal solution. Given an input sentence, our model deconstructs it into component phrases and clauses, each of which is simplified (lexically and structurally) through QG rewrite rules. We generate all possible simplifications for a given input and use the ILP to find the best target subject to grammaticality constraints. In what follows we first detail how we extract QG rewrite rules as these form the backbone of our model and then formulate the ILP proper.

3.1 Quasi-synchronous Grammar

Phrase alignment Our model operates on individual sentences annotated with syntactic information i.e., phrase structure trees. In our experiments, we obtain this information from the Stanford parser (Klein and Manning, 2003) but any other broadly similar parser could be used instead. Given an input sentence S1 or its parse tree T1, the QG constructs a monolingual grammar for parsing, or generating, possible translation trees T2. A grammar node in the target tree T2 is modeled on a subset of nodes in the source tree, with a rather loose alignment between the trees.

We take aligned sentence pairs represented as phrase structure trees and build up a list of leaf node alignments based on lexical identity. We align direct parent nodes where more than one child node aligns. QG rules are created from aligned nodes above the leaf node level if the all the nodes in the target tree can be explained using nodes from the source. This helps to improve the quality in what is inherently a noisy process, and it is largely responsible for a relatively small resulting grammar (see Table 2). Examples of phrase alignments (indicated with dotted lines) are shown in Figure 1.

Syntactic simplification rules Each QG rule describes the transformations required from source to target phrase sub-trees. It allows child (and possibly grand-child) constituents to be deleted or re-



Rule involving lexical substitution: $\langle$VP, VP$\rangle \rightarrow \langle$[ADVP [RB *afterwards*] $\text{VBD}_{\boxed{3}}$ $\text{NP}_{\boxed{4}}$], [$\text{VBD}_{\boxed{3}}$ $\text{NP}_{\boxed{4}}$ ADVP [RB *later*]]$\rangle$
Rule for splitting into main constituent and auxiliary sentence: $\langle$VP, VP, ST$\rangle \rightarrow \langle$[$\text{VP}_{\boxed{1}}$ *and* $\text{VP}_{\boxed{2}}$], [$\text{VP}_{\boxed{1}}$], [NP [PRP *He*] $\text{VP}_{\boxed{2}}$.]$\rangle$

Figure 1: A source sentence (upper tree) is split into two sentences. Dotted lines show word alignments, while boxed subscripts show aligned nodes used to form QG rules. Below, two QG rules learned from this data.

ordered, and for nodes to be flattened. In addition, we allow insertion of punctuation and some function words, identified by a small set of POS tags. To distinguish sentences proper (which have final punctuation) from clauses, we modify the output of the parser, changing the root sentence parse tag from S to ST (a "top-level sentence"); this allows clauses to be extracted and rewritten as stand-alone sentences.

Lexical simplification rules Lexical substitutions are an important part of simplification. We learn them from aligned sub-trees, in the same way as described above for syntax rules, by allowing a small number of lexical substitutions to be present in the rules, and provided they do not include proper nouns. The resulting QG rules could be applied by matching the syntax of the whole sub-tree surrounding the substitution, but this approach is overly restrictive and suffers from data sparsity. Indeed, Yatskar et al. (2010) learn lexical simplifications without taking syntactic context into account. We therefore add a post-processing stage to the learning process. For rules where the syntactic structures of the source and target sub-trees match, and the only difference is a lexical substitution, we construct a more general rule by extracting the words and corresponding POS tags involved in the substitution. Then at the generation stage, identifying suitable rules depends only on the substitution words, rather than the surrounding syntactic context. An example of a lexical substitution rule is shown in Figure 1.

Sentence splitting rules Another important simplification technique is to split syntactically complicated sentences into several shorter ones. To learn QG rules for this operation, the source sentence is aligned with two consecutive target sentences.

Rather than expecting to discover a split point in the source sentence, we attempt to identify a node in the source parse tree that contributes to both of the two target sentences. Our intuition is that one of the target sentences will follow the general syntactic structure of the source sentence. We designate this as the *main* sentence. A node in the source sentence parse tree will be aligned with a (similar but simpler) node in the main target sentence, but at the same time it will fully explain the other target sentence, which we term the *auxiliary* sentence. It is

possible for the auxiliary sentence to come before or after the main sentence. In the learning procedure, we try both possible orderings, and record the order in any QG rules successfully produced.

The resulting QG rule is a tuple of three phrase structure elements: the source node, the node in the target main sentence (the top level of this node is typically the same as that of the source node), and the phrase structure of the entire auxiliary sentence.[1] In addition, there is a flag to indicate if the auxiliary sentence comes before or after the main sentence. This formalism is able to capture the operations required to split sentences containing coordinate or subordinate clauses, parenthetical content, relative clauses and apposition. An example of a sentence splitting rule is illustrated in Figure 1.

3.2 ILP-based Generation

We cast the problem of finding a suitable target simplification given a source sentence as an integer linear program (ILP). Specifically, simplified text is created from source sentence parse trees by identifying and applying QG grammar rules. These will have matching structure and may also require lexical matching (shown using italics in the example rules in Figure 1). The generation process starts at the root node of the parse tree, applying QG rules to subtrees until leaf nodes are reached. We do not use the Bayesian probability model proposed by Smith and Eisner (2006) to identify the best sequence of simplification rules. Instead, where there is more than one matching rule, and so more than one simplification is possible, the alternatives are all generated and incorporated into the target phrase structure tree. The ILP model operates over this phrase structure tree and selects the phrase nodes from which to form the target output.

Applying the QG rules on the source sentence generates a number of auxiliary sentences. Let $\mathcal{S}$ be this set of sentences. Let $\mathcal{P}$ be the set of nodes in the phrase structure trees of the auxiliary sentences, and $\mathcal{P}_s \subset \mathcal{P}$ be the set of nodes in each sentence $s \in \mathcal{S}$. Let the sets $\mathcal{D}_i \subset \mathcal{P}, \forall i \in \mathcal{P}$ capture the phrase dependency information for each node i, where each set $\mathcal{D}_i$ contains the nodes that depend on the presence of i. In a similar fashion, the sets $\mathcal{A}_i \subset \mathcal{S}, \forall i \in \mathcal{P}$ capture the indices of any auxiliary sentences that depend on the presence of node i. $\mathcal{C} \subset \mathcal{P}$ is the set of nodes involving a choice of alternative simplifications (nodes in the tree where more than one QG rewrite rule can be applied, as mentioned above); $\mathcal{C}_i \subset \mathcal{P}, i \in \mathcal{C}$ are the sets of nodes that are direct children of each such node, in other words they are the individual simplifications. Let $l_i^{(w)}$ be the length of each node i in words, and $l_i^{(sy)}$ its length in syllables. As we shall see below counts of words and syllables are important cues in assessing readability.

The model is cast as an binary integer linear program. A vector of binary decision variables $x \in \{0,1\}^{|\mathcal{P}|}$ indicates if each node is to be part of the output. A vector of auxiliary binary variables $y \in \{0,1\}^{|\mathcal{S}|}$ indicates which (auxiliary) sentences have been chosen.

$$\max_{x} \quad \sum_{i \in \mathcal{P}} g_i x_i + h_w + h_{sy} \tag{1a}$$

$$\text{s.t.} \quad x_j \rightarrow x_i \qquad \forall i \in \mathcal{P}, j \in \mathcal{D}_i \tag{1b}$$

$$x_i \rightarrow y_s \qquad \forall i \in \mathcal{P}, s \in \mathcal{A}_i \tag{1c}$$

$$x_i \rightarrow y_s \qquad \forall s \in \mathcal{S}, i \in \mathcal{P}_s \tag{1d}$$

$$\sum_{j \in \mathcal{C}_i} x_j = x_i \qquad \forall i \in \mathcal{C}, j \in \mathcal{C}_i \tag{1e}$$

$$\sum_{s \in \mathcal{S}} y_i \geq 1 \tag{1f}$$

$$x_i \in \{0,1\} \qquad \forall i \in \mathcal{P} \tag{1g}$$

$$y_s \in \{0,1\} \qquad \forall s \in \mathcal{S}. \tag{1h}$$

Our objective function, given in Equation (1a), is the summation of local and global components. Each phrase is locally given a *rewrite penalty* g_i, where common lexical substitutions, rewrites and simplifications are penalized less (as we trust them more), compared to rarer QG rules. The penalty is a simple log-probability measure, $g_i = \log\left(\frac{n_r}{N_r}\right)$, where n_r is the number of times the QG rule r was seen in the training data, and N_r the number of times all suitable rules for this phrase node were seen. If no suitable rules exist, we set $g_i = 0$.

The other two components of the objective, h_w and h_{sy}, are global in nature, and guide the ILP

[1] Note that the target component comprises the second and third elements as a pair, and variables from the source component are split between them.

towards simpler language. They draw inspiration from existing measures of readability (the ease with which a document can be read and understood). The primary aim of readability formulas is to assess whether texts or books are suitable for students at particular grade levels or ages (see Mitchell 1985 for an overview). Intuitively, texts ought to be simpler if they correspond to low reading levels. A commonly used reading level measure is the Flesch-Kincaid Grade Level (FKGL) index which estimates readability as a combination of the average number of syllables per word and the average number of words per sentence. Unfortunately, this measure is non-linear[2] and cannot be incorporated directly into the objective of the ILP. Instead, we propose a linear approximation. We provide the ILP with targets for the average number of words per sentence (wps), and syllables per word (spw). $h_w(x,y)$ then measures the number of words below this target level that the ILP has achieved:

$$h_w(x,y) = \text{wps} \times \sum_{i \in \mathcal{S}} y_i - \sum_{i \in \mathcal{P}} l_i^{(w)} x_i.$$

When positive, this indicates that sentences are shorter than target, and contributes positively to the readability objective whilst encouraging the application of sentence splitting and deletion-based QG rules. Similarly, $h_{sy}(x,y)$ measures the number of syllables below that expected, from the target average and the number of words the ILP has chosen:

$$h_{sy}(x) = \text{spw} \times \sum_{i \in \mathcal{P}} l_i^{(w)} x_i - \sum_{i \in \mathcal{P}} l_i^{(sy)} x_i.$$

This component of the objective encourages the deletion or lexical substitution of complex words. We can use the two target parameters (wps and spw) to control how much simplification the ILP should apply.

Constraint (1b) enforces grammatical correctness by ensuring that the phrase dependencies are respected and the resulting structure is a tree. Phrases that depend on phrase i are contained in the set $\mathcal{D}_i$. Variable x_i is true, and therefore phrase i will be included in the target output, if any of its dependents $x_j \in \mathcal{D}_i$ are true.[3] Constraint (1c) links main phrases to auxiliary sentences, so that the latter can only be included in the output if the main phrase has also been chosen. This helps to control coherence within the output text. Despite seeming similar to (1c), the role of constraint (1d) is quite different. It links phrase variables x to sentence variables y, to ensure the logical integrity of the model is correct. Where the QG provides alternative simplifications, it makes sense of course to select only one. This is controlled by constraint (1e), and by placing all alternatives in the set $\mathcal{D}_i$ for the node i.

With these constraints alone, and faced with a source sentence that is particularly difficult to simplify, it is possible for the ILP solver to return a "trivial" solution of no output at all, as all other available solutions result in a negative objective value. It is therefore necessary to impose a global minimum output constraint (1f). In combination with the dependency relations in (1c), this constraint ensures that at least an element of the root sentence is present in the output. Global maximum length constraints are a frequently occurring aspect of ILP models used in NLP applications. We decided not to incorporate any such constraints into our model, as we did not want to place limitations on the simplification of original content.

4 Experimental Setup

In this section we present our experimental setup for assessing the performance of the simplification model described above. We give details on the corpora and grammars we used, model parameters, the systems used for comparison with our approach, and explain how the output was evaluated.

Grammar Extraction QG rules were learned from revision histories and an aligned simplification corpus, which we obtained from snapshots[4] of MainEW and SimpleEW. Wiki-related mark-up and meta-information was removed to extract the plain text from the articles.

SimpleEW revisions not only simplify the text of existing articles, they may also introduce new content, vandalize or remove vandalism, or perform numerous automatic "house-keeping" modifications.

[2] $\text{FKGL} = 0.39\left(\frac{\text{total words}}{\text{total sentences}}\right) + 1.8\left(\frac{\text{total syllables}}{\text{total words}}\right) - 15.59$

[3] Constraints (1b), (1c) and (1d) are shown as dependencies for clarity, but they were implemented as inequalities in the ILP.

[4] The snapshots for MainEW (enwiki) and SimpleEW (simplewiki dated 2010-09-16 and 2010-09-13, respectively (both available from `http://download.wikimedia.org/`).

Corpora	Syntactic	Lexical	Splitting
Revision	316	269	184
Aligned	312	96	254

Table 2: Number of QG rules extracted (after removing singletons) from revision-based and aligned corpora.

We identified suitable revisions for simplification by selecting those where the author had mentioned a keyword (such as *simple*, *clarification* or *grammar*) in the revision comments. Each selected revision was compared to the previous version. Because the entire article is stored at each revision, we needed to identify and align modified sentences. We first identified modified sections using the Unix `diff` program, and then individual sentences within the sections were aligned using the program `dwdiff`[5]. This resulted in 14,831 paired sentences. With regard to the aligned simplification corpus, we paired 15,000 articles from SimpleEW and MainEW following the language link within the snapshot files. Within the paired articles, we identified aligned sentences using macro alignment (at paragraph level) then micro alignment (at sentence level), using tf.idf scores to measure similarity (Barzilay and Elhadad, 2003; Nelken and Schieber, 2006).

All source-target sentences (resulting from revisions or alignments) were parsed with the Stanford parser (Klein and Manning, 2003) in order to label the text with syntactic information. QG rules were created by aligning nodes in these sentences as described earlier. A breakdown of the number and type of rules we obtained from the revision and aligned corpora (after removing rules appearing only once) is given in Table 2. Examples of the most frequently learned QG rules are shown in Table 3. Rules (1)–(3) involve syntactic simplification and rules (4)–(6) involve sentence splitting. Examples of common lexical simplifications found by our grammar are: "*discovered*" → "*found*", "*defeated*" → "*won against*", "*may refer to*" → "*could mean*", "*original*" → "*first*", "*requires*" → "*needs*".

Sentence generation We generated simplified versions of MainEW sentences. For each (parsed) source sentence, we created and solved an ILP (see Equation (1)) parametrized as follows: the number of target words per sentence (wps) was set to 8, and syllables per word (spw) to 1.5. These two parameters were empirically tuned on the training set. To solve the ILP model we used the ZIB Optimization Suite software (Achterberg, 2007; Koch, 2004). The solution was converted into a sentence by removing nodes not chosen from the tree representation, then concatenating the remaining leaf nodes in order.

1.	$\langle \text{S, ST} \rangle \rightarrow \langle [\text{NP}_{\boxed{1}}\ \text{VP}_{\boxed{2}}], [\text{NP}_{\boxed{1}}\ \text{VP}_{\boxed{2}}\ .] \rangle$
2.	$\langle \text{S, ST} \rangle \rightarrow \langle [\text{VP}_{\boxed{1}}], [\textit{This}\ \text{VP}_{\boxed{1}}\ .] \rangle$
3.	$\langle \text{NP, ST} \rangle \rightarrow \langle [\text{NP}_{\boxed{1}}, \text{NP}_{\boxed{2}}], [\text{NP}_{\boxed{1}}\ \textit{was}\ \text{VP}_{\boxed{2}}\ .] \rangle$
4.	$\langle \text{ST, ST, ST} \rangle \rightarrow \langle [\text{S}_{\boxed{1}}\ \textit{, and}\ \text{S}_{\boxed{2}}], [\text{ST}_{\boxed{1}}], [\text{ST}_{\boxed{2}}] \rangle$
5.	$\langle \text{ST, ST, ST} \rangle \rightarrow \langle [\text{S}_{\boxed{1}}\ \textit{:}\ \text{S}_{\boxed{2}}], [\text{ST}_{\boxed{1}}], [\text{ST}_{\boxed{2}}] \rangle$
6.	$\langle \text{ST, ST, ST} \rangle \rightarrow \langle [\text{S}_{\boxed{1}}\ \textit{, but}\ \text{S}_{\boxed{2}}], [\text{ST}_{\boxed{1}}], [\text{ST}_{\boxed{2}}] \rangle$

Table 3: Examples of QG rules involving syntactic simplification (1)–(3) and sentence division (4)–(6). The latter are shown as the tuple ⟨source, target, aux⟩. The transform of nodes from S to ST (for example) rely on the application of syntactic simplification rules rules. Boxed subscripts show aligned nodes.

Evaluation We evaluated our model on the same dataset used in Zhu et al. (2010), an aligned corpus of MainEW and SimpleEW sentences. The corpus contains 100/131 source/target sentences and was created automatically. Sentences from this corpus (and their revisions) were excluded from training. We evaluated two versions of our model, one with rewrite rules acquired from revision histories of simplified documents and another one with rules extracted from MainEW-SimpleEW aligned sentences. These models were compared against Zhu et al. (2010)[6] who also learn simplification rules from Wikipedia, and a simple baseline that uses solely lexical simplifications[7] provided by the SimpleEW editor "SpencerK" (Spencer Kelly). An obvious idea would be to treat sentence simplification as an English-to-English translation problem and use an off-the-shelf system like Moses[8] for the task. However, we refrained from doing so as Zhu et al. (2010) show that Moses performs poorly, it cannot model rewrite operations that split sentences or drop words and in most cases generates output identical

[5] `http://os.ghalkes.nl/dwdiff.html`

[6] We are grateful to Zhemin Zhu for providing us with his test set and the output of his system.

[7] `http://www.spencerwaterbed.com/soft/simple/`

[8] `http://www.statmt.org/moses/`

MainEW	Wonder has recorded several critically acclaimed albums and hit singles, and writes and produces songs for many of his label mates and outside artists as well.
Zhu et al	Wonder has recorded several praised albums and writes and produces songs. Many of his label mates and outside artists as well.
AlignILP	Wonder has recorded several critically acclaimed albums and hit singles. He produces songs for many of his label mates and outside artists as well. He writes.
RevILP	Wonder has recorded many critically acclaimed albums and hit singles. He writes. He makes songs for many of his label mates and outside artists as well.
SimpleEW	He has recorded 23 albums and many hit singles, and written and produced songs for many of his label mates and other artists as well.
MainEW	The London journeys In 1790, Prince Nikolaus died and was succeeded by a thoroughly unmusical prince who dismissed the entire musical establishment and put Haydn on a pension.
Zhu et al	The London journeys in 1790, prince Nikolaus died and was succeeds by a son became prince. A son became prince told the entire musical start and put he on a pension.
AlignILP	The London journeys In 1790, Prince Nikolaus died. He was succeeded by a thoroughly unmusical prince. He dismissed the entire musical establishment. He put Haydn on a pension.
RevILP	The London journeys In 1790, Prince Nikolaus died. He was succeeded by a thoroughly unmusical prince. He dismissed the whole musical establishment. He put Haydn on a pension.
SimpleEW	The London journeys In 1790, Prince Nikolaus died and his son became prince. Haydn was put on a pension.

Table 4: Example simplifications produced by the systems in this paper (RevILP, AlignILP) and Zhu et al.'s (2010) model, compared to real Wikipedia text (MainEW: input source, SimpleEW: simplified target).

to the source.

We evaluated model output in two ways, using automatic evaluation measures and human judgments. Intuitively, readability measures ought to be suitable for assessing the output of simplification systems. We report results with the well-known Flesch-Kincaid Grade Level index (FKGL). Experiments with other readability measures such as the Flesch Reading Ease and the Coleman-Liau index obtained similar results. In addition, we also assessed how the system output differed from the human SimpleEW gold standard by computing BLEU (Papineni et al., 2002) and TERp (Snover et al., 2009). Both measures are commonly used to automatically evaluate the quality of machine translation output. BLEU[9] scores the target output by counting n-gram matches with the reference, whereas TERp is similar to word error rate, the only difference being that it allows shifts and thus can account for word order differences. TERp also allows for stem, synonym, and paraphrase substitutions which are common rewrite operations in simplification.

In line with previous work on text rewriting (e.g., Knight and Marcu 2002) we also evaluated system output by eliciting human judgments. We conducted three experiments. In the first experiment participants were presented with a source sentence and its target simplification and asked to rate whether the latter was easier to read compared to the source. In the second experiment, they were asked to rate the grammaticality of the simplified output. In the third experiment, they judged how well the simplification preserved the meaning of the source. In all experiments participants used a five point rating scale where a high number indicates better performance. We randomly selected and automatically simplified 64 sentences from Zhu et al.'s (2010) test corpus using the four models described above. We also included gold standard simplifications. Our materials thus consisted of 320 (64×5) source-target sentences.[10] We collected ratings from 45 unpaid volunteers, all self reported native English speakers. The studies were conducted over the Internet using a custom built web interface. Examples of our experimental items are given in Table 4 (we omit the output of SpencerK as this is broadly similar to the source sentence, modulo lexical substitutions).

[9] We calculated single-reference BLEU using the *mteval-v13a* script (with the default settings).

[10] A Latin square design ensured that subjects did not see two different simplifications of the same sentence.

Models	FKGL	BLEU	TERP
MainEW	15.12	—	—
SimpleEW	11.25	—	—
SpencerK	14.67	0.47	0.51
Zhu et al	9.41	0.38	0.59
RevILP	10.92	0.42	0.60
AlignILP	12.36	0.34	0.85

Table 5: Model performance using automatic evaluation measures.

5 Results

The results of our automatic evaluation are summarized in Table 5. The first column reports the FKGL readability index of the source sentences (MainEW), of their target simplifications (SimpleEW) and the output of four models: a simple baseline that relies on lexical substitution (SpencerK), Zhu et al.'s (2010) model, and two versions of our model, one trained on revision histories (RevILP) and another one trained on the MainEW-SimpleEW aligned corpus (AlignILP). As can be seen, the source sentences have the highest reading level. Zhu et al.'s system has the lowest reading level followed by our own models and SpencerK. All models are significantly[11] different in reading level from SimpleEW with the exception of RevILP (using a one-way ANOVA with post-hoc Tukey HSD tests). SpencerK is not significantly different in readability from MainEW; RevILP is significantly different from Zhu et al. and AlignILP. In sum, these results indicate that RevILP is the closest to SimpleEW and that the provenance of the QG rules has an impact on the model's performance.

Table 5 also shows BLEU and TERp scores with SimpleEW as the reference. These scores can be used to examine how close to the gold standard our models are. SpencerK has the highest BLEU and lowest TERp scores.[12] This is expected as this baseline performs only a very limited type of rewriting, namely lexical substitution. AlignILP is most different from the reference, followed by Zhu et al. (2010) and RevILP. Taken together these results indicate that the ILP models perform a fair amount of rewriting without simply rehashing the source sentence.

[11] All significance differences reported throughout this paper are with a level less than 0.01.

[12] The perfect BLEU score is one and the perfect TERp score is zero.

Models	Simplicity	Grammaticality	Meaning
SimpleEW	3.74	4.89	4.41
SpencerK	1.41	4.87	4.84
Zhu et al	2.92	3.43	3.44
RevILP	3.64	4.55	4.19
AlignILP	2.69	4.03	3.98

Table 6: Average human ratings for gold standard SimpleEW sentences, a simple baseline (SpencerK) based on lexical substitution, Zhu et al.'s 2010 model, and two versions of our ILP model (RevILP and AlignILP).

	Zhu et al	AlignILP	RevILP	SimpleEW
SpencerK	□◇△	□◇△	□⧫△	□⧫△
Zhu et al		■◇△	□◇△	□◇△
AlignILP			□◇▲	□◇△
RevILP				■⧫▲

Table 7: □/■: is/not sig. diff. wrt simplicity; ◇/⧫: is/not sig. diff. wrt grammaticality; △/▲: is/not sig. diff. wrt meaning.

We now turn to the results of our judgment elicitation study. Table 6 reports the average ratings for Simplicity (is the target sentence simpler than the source?), Grammaticality (is the target sentence grammatical?), and Meaning (does the target preserve the meaning of the source?). With regard to simplicity, our participants perceive the gold standard (SimpleEW) to be the simplest, followed by RevILP, Zhu et al, and AlignILP. SpencerK is the least simple model and the most grammatical one as lexical substitutions do not change the structure of the sentence. Interestingly, RevILP and AlignILP are also rated highly with regard to grammaticality. Zhu et al. (2010) is the least grammatical model. Finally, RevILP preserves the meaning of the target as well as SimpleEW, whereas Zhu et al. yields the most distortions. Again SpencerK is rated highly amongst the other models as it is does not substantially simplify and thus change the meaning of the source.

Table 7 reports on pairwise comparisons between all models and their statistical significance (again using a one-way ANOVA with post-hoc Tukey HSD tests). RevILP is not significantly different from SimpleEW on any dimension (Simplicity, Grammat-

Original story: There was once a sweet little maid *who* lived with her father and mother in a pretty little cottage at the edge of the village. At the further end of the wood *was another pretty cottage and in it* lived her grandmother. Everybody loved this little *girl,* her grandmother perhaps loved her most of all *and gave* her a great many pretty things. Once she gave her a red cloak with a hood *which she always wore,* so *people* called her Little Red Riding Hood.
Generated simplification: There was once a sweet little maid. She lived with her father and mother in a pretty little cottage at the edge of the village. At the further end of the wood it lived her grandmother. Everybody loved this little girl. Her grandmother perhaps loved her most of all. She gave her a great many pretty things. Once she gave her a red cloak with a hood, so persons called her Little Red Riding Hood.

Table 8: Excerpt of *Little Red Riding Hood* simplified by the RevILP model. Modifications to the original story are highlighted in italics.

icality, Meaning), whereas Zhu et al. differs significantly from RevILP and SimpleEW on all dimensions. It is also significantly different from AlignILP in terms of grammaticality and meaning but not simplicity. RevILP is significantly more simple and grammatical than AlignILP but performs comparably with respect to preserving the meaning of the source.

In sum, our results show that RevILP is the best performing model. It creates sentences that are simple, grammatical and adhere to the meaning of the source. The QG rules obtained from the revision histories produce better output compared to the aligned corpus. As revision histories are created by Wikipedia contributors, they tend to be a more accurate data source than aligned sentences which are obtained via an automatic and unavoidably noisy procedure. Our results also show that a more general model not restricted to specific rewrite operations like Zhu et al. (2010) obtains superior results and has better coverage.

We also wanted to see whether a simplification model trained on Wikipedia could be applied to another domain. To this end, we used RevILP to simplify five children stories from the Gutenburg[13] collection. The model simplified one sentence at a time and was ran with the Wikipedia settings without any modification. The mean FKGL on the simplified stories was 3.78. compared to 7.04 for the original ones. An example of our system's output on *Little Red Riding Hood* is shown in Table 8.

Possible extensions and improvements to the current model are many and varied. We have presented an all-purpose simplification model without a target audience or application in mind. An interesting research direction would be to simplify text according to readability levels or text genres (e.g., newspaper vs literary text). We could do this by incorporating readability-specific constraints to the ILP or by changing the objective function (e.g., by favoring more domain-specific rules). Finally, we would like to extend the current model so as to simplify entire documents both in terms of style and content.

Acknowledgments We are grateful to Lillian Lee whose invited talk at CoNLL-2010 inspired this research. We would also like to thank the members of the Probabilistic Models of Language group at the School of Informatics for valuable discussions and comments. We acknowledge the support of EPSRC through project grant EP/F055765/1.

References

Achterberg, Tobias. 2007. *Constraint Integer Programming*. Ph.D. thesis, Technische Universität Berlin.

Barzilay, Regina. 2003. *Information Fusion for Multi-Document Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University.

Barzilay, Regina and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Sapporo, Japan, pages 25–32.

Beigman Klebanov, Beata, Kevin Knight, and Daniel Marcu. 2004. Text simplification for information-seeking applications. In *Proceedings of Ontologies, Dabases, and Applications of Semantics (ODBASE) International Conference*.

[13] http://www.gutenberg.org

Springer, Agia Napa, Cyprus, volume 3290 of *Lecture Notes in Computer Science*, pages 735–747.

Carroll, John, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the ACL*. Bergen, Norway, pages 269–270.

Chandrasekar, Raman, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th International Conference on Computational Linguistics*. Copenhagen, Denmark, pages 1041–1044.

Cohn, Trevor and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics*. Manchester, UK, pages 137–144.

Das, Dipanjan and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the ACL-IJCNLP*. Suntec, Singapore, pages 468–476.

Devlin, Siobhan. 1999. *Simplifying Natural Language for Aphasic Readers*. Ph.D. thesis, University of Sunderland.

Inui, Kentaro, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: A project note. In *Proceedings of the Second International Workshop on Paraphrasing*. Association for Computational Linguistics, Sapporo, Japan, pages 9–16.

Kaji, Nobuhiro, Daisuke Kawahara, Sadao Kurohashi, and Satoshi Sato. 2002. Verb paraphrase based on case frame alignment. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 215–222.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics*. Sapporo, Japan, pages 423–430.

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

Koch, Thorsten. 2004. *Rapid Mathematical Prototyping*. Ph.D. thesis, Technische Universität Berlin.

Mitchell, James V. 1985. *The Ninth Mental Measurements Year-book*. University of Nebraska Press, Lincoln, Nebraska.

Nastase, Vivi and Michael Strube. 2008. Decoding Wikipedia categories for knowledge acquisition. In *Proceedings of the 23rd Conference on Artificial Intelligence*. pages 1219–1224.

Nelken, Rani and Stuart Schieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy, pages 161–168.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*. Philadelphia, PA, pages 311–318.

Ponzetto, Simone Paolo and Michael Strube. 2007. Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research* 30:181–212.

Sauper, Christina and Regina Barzilay. 2009. Automatically generating Wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 208–216.

Siddharthan, Advaith. 2003. *Syntactic Simplification and Text Cohesion*. Ph.D. thesis, University of Cambridge, University of Cambridge.

Siddharthan, Advaith. 2004. Syntactic simplification and text cohesion. in research on language and computation. *Research on Language and Computation* 4(1):77–109.

Smith, David and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings on the Workshop on Statistical Machine Transla-*

tion. Association for Computational Linguistics, New York City, pages 23–30.

Smith, David A. and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the EMNLP*. Suntec, Singapore, pages 822–831.

Snover, Matthew, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Athens, Greece, pages 259–268.

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, pages 344–352.

Wang, Mengqiu, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the EMNLP-CoNLL*. Prague, Czech Republic, pages 22–32.

Watanabe, Willian Massami, Arnaldo Candido Junior, Vinícius Rodriguez de Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Alu´sio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM International Conference on Design of Communication*. Bloomington, IN.

Woodsend, Kristian, Yansong Feng, and Mirella Lapata. 2010. Title generation with quasi-synchronous grammar. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 513–523.

Wu, Fei and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 118–127.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*. Toulouse, France, pages 523–530.

Yamangil, Elif and Rani Nelken. 2008. Mining Wikipedia revision histories for improving sentence compression. In *Proceedings of ACL-08: HLT, Short Papers*. Association for Computational Linguistics, Columbus, Ohio, pages 137–140.

Yatskar, Mark, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from Wikipedia. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*. pages 365–368.

Zhao, Shiqi, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Singapore, pages 834–842.

Zhu, Zhemin, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Beijing, China, pages 1353–1361.

Bootstrapping Semantic Parsers from Conversations

Yoav Artzi and **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{yoav,lsz}@cs.washington.edu

Abstract

Conversations provide rich opportunities for interactive, continuous learning. When something goes wrong, a system can ask for clarification, rewording, or otherwise redirect the interaction to achieve its goals. In this paper, we present an approach for using conversational interactions of this type to induce semantic parsers. We demonstrate learning without any explicit annotation of the meanings of user utterances. Instead, we model meaning with latent variables, and introduce a loss function to measure how well potential meanings match the conversation. This loss drives the overall learning approach, which induces a weighted CCG grammar that could be used to automatically bootstrap the semantic analysis component in a complete dialog system. Experiments on DARPA Communicator conversational logs demonstrate effective learning, despite requiring no explicit meaning annotations.

1 Introduction

Conversational interactions provide significant opportunities for autonomous learning. A well-defined goal allows a system to engage in remediations when confused, such as asking for clarification, rewording, or additional explanation. The user's response to such requests provides a strong, if often indirect, signal that can be used to learn to avoid the original confusion in future conversations. In this paper, we show how to use this type of conversational feedback to learn to better recover the meaning of user utterances, by inducing semantic parsers from unannotated conversational logs. We believe that this style of learning will contribute to the long term goal of building self-improving dialog systems that continually learn from their mistakes, with little or no human intervention.

Many dialog systems use a semantic parsing component to analyze user utterances (e.g., Allen et al., 2007; Litman et al., 2009; Young et al., 2010). For example, in a flight booking system, the sentence

Sent: I want to go to Seattle on Friday
LF: $\lambda x.to(x, SEA) \wedge date(x, FRI)$

might be mapped to the logical form (LF) meaning representation above, a lambda-calculus expression defining the set of flights that match the user's desired constraints. This LF is a representation of the semantic content that comes from the sentence, and would be input to a context-dependent understanding component in a full dialog system, for example to find the date that the symbol FRI refers to.

To induce semantic parsers from interactions, we consider user statements in conversational logs and model their meaning with latent variables. We demonstrate that it is often possible to use the dialog that follows a statement (including remediations such as clarifications, simplifications, etc.) to learn the meaning of the original sentence. For example, consider the first user utterance in Figure 1, where the system failed to understand the user's request. To complete the task, the system must use a remediation strategy. Here, it takes the initiative by asking for and confirming each flight constraint in turn. This strategy produces an unnatural conversation but provides supervision for learning the meaning of the

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 421–432,
Edinburgh, Scotland, UK, July 27–31, 2011.

original utterance. We can easily record representations of the meanings the system intended to convey at each step, as seen in Figure 1, and use this indirect supervision for learning.

Learning from this weak signal is challenging. In any specific conversation, the system's remediations can fail to recover aspects of the original user meaning and can introduce spurious constraints, for example when users change their goals mid conversation. To learn effectively, the model must accumulate evidence from many interactions to best recover the meaning of each specific sentence.

We will learn semantic parsers defined by probabilistic Combinatory Categorial Grammars (PCCGs), which include both a lexicon and a weighted linear model for parse selection. The lexicon specifies the meanings of individual words and phrases, while the parameters of a parsing model define how to best combine word- and phrase-level meanings to analyze complete sentences. To learn without labeled meaning representations, we make use of a variant of the loss-sensitive Perceptron algorithm (Singh-Miller and Collins, 2007). We define loss functions to provide a rough measure of (1) how well a candidate meaning for a utterance matches the conversation that follows it and (2) how well the candidate matches our expectations about the types of things that are often said in the dialog's domain. These notions of loss drive not only the parameter estimation but also the grammar induction process that constructs the CCG lexicon. Experiments on conversation logs from the DARPA Communicator corpus (Walker et al., 2002) demonstrate the feasibility of our approach.

This paper makes the following contributions:

- A formalization of the problem of learning the meaning of user statements from conversational feedback, without requiring annotation.
- A new loss-sensitive learning algorithm for this problem that induces semantic parsers from conversation logs.
- Loss functions to measure the quality of hypothetical utterance meanings within the conversation in which they appear.
- An evaluation on logs from two dialog systems that demonstrate effective learning from conversations alone.

SYSTEM: how can i help you? (OPEN_TASK)
USER: i would like to fly from atlanta georgia to london england on september twenty fourth in the early evening i would like to return on october first departing from london in the late morning
SYSTEM: leaving what city? (ASK:$\lambda x.from(fl, x)$)
USER: atlanta georgia
SYSTEM: leaving atlanta. (CONFIRM:$from(fl, ATL)$) going to which city? (ASK:$\lambda x.to(fl, x)$)
USER: london
SYSTEM: arriving in london england. (CONFIRM:$to(fl, LON)$) what date would you like to depart atlanta? (ASK:$\lambda x.from(fl, ATL) \wedge depart\text{-}date(fl, x)$)
USER: september twenty fourth in the early evening
[conversation continues]

Figure 1: Conversational excerpt from a DARPA Communicator travel-planning dialog. Each system statement is labeled with representations of its speech act and logical meaning, in parentheses. The user utterances have no labels. Conversations of this type provide the training data to learn semantic parsers for user utterances.

2 Problem

Our goal is to learn a function that maps a sentence x to a lambda-calculus expression z. We assume access to logs of conversations with automatically generated annotation of system utterance meanings, but no explicit labeling of each user utterance meaning.

We define a conversation $\mathcal{C} = (\vec{U}, O)$ to be a sequence of utterances $\vec{U} = [u_0, \ldots, u_m]$ and a set of conversational objects O. An object $o \in O$ is an entity that is being discussed, for example there would be a unique object for each flight leg discussed in a travel planning conversation. Each utterance $u_i = (s, x, a, z)$ represents the speaker $s \in \{User, System\}$ producing the natural language statement x which asserts a speech act $a \in \{ASK, CONFIRM, \ldots\}$ with meaning representation z. For example, from the second system utterance in Figure 1 the question $x =$"Leaving what city?" is an a=ASK speech act with lambda-calculus meaning $z = \lambda x.from(fl, x)$. This meaning represents the fact that the system asked for the departure city for the conversational object $o = fl$ representing the flight leg that is currently being discussed. We will learn from conversations where the speech

acts a and logical forms z for user utterances are unlabeled. Such data can be generated by recording interactions, along with each system's internal representation of its own utterances.

Finally, since we will be analyzing sentences at a specific point in a complete conversation, we define our training data as a set $\{(j_i, \mathcal{C}_i)|i = 1 \ldots n\}$. Each pair is a conversation $\mathcal{C}_i$ and the index j_i of the user utterance x in $\mathcal{C}_i$ whose meaning we will attempt to learn to recover. In general, the same conversation $\mathcal{C}$ can be used in multiple examples, each with a different sentence index. Section 8 provides the details of how the data was gathered for our experiments.

3 Overview of Approach

We will present an algorithm for learning a weighted CCG parser, as defined in Section 5, that can be used to map sentences to logical forms. The approach induces a lexicon to represent the meanings of words and phrases while also estimating the parameters of a weighted linear model for selecting the best parse given the lexicon.

Learning As defined in Section 2, the algorithm takes a set of n training examples, $\{(j_i, \mathcal{C}_i) : i = 1, \ldots, n\}$. For each example, our goal is to learn to parse the user utterance x at position j_i in $\mathcal{C}_i$. The training data contains no direct evidence about the logical form z that should be paired with x, or the CCG analysis that would be used to construct z. We model all of these choices as latent variables.

To learn effectively in this complex, latent space, we introduce a loss function $\mathcal{L}(z, j, \mathcal{C}) \in \mathbb{R}$ that measures how well a logical form z models the meaning for the user utterance at position j in $\mathcal{C}$. In Section 6, we will present the details of the loss we use, which is designed to be sensitive to remediations in $\mathcal{C}$ (system requests for clarification, etc.) but also be robust to the fact that conversations often do not uniquely determine which z should be selected, for example when the user prematurely ends the discussion. Then, in Section 7, we present an approach for incorporating this loss function into a complete algorithm that induces a CCG lexicon and estimates the parameters of the parsing model.

This learning setup focuses on a subproblem in dialog; semantic interpretation. We do not yet learn to recover user speech acts or integrate the logical form into the context of the conversation. These are important areas for future work.

Evaluation We will evaluate performance on a test set $\{(x_i, z_i)|i = 1, \ldots, m\}$ of m sentences x_i that have been explicitly labeled with logical forms z_i. This data will allow us to directly evaluate the quality of the learned model. Each sentence is analyzed with the learned model alone; the loss function and any conversational context are not used during evaluation. Parsers that perform well in this setting will be strong candidates for inclusion in a more complete dialog system, as motivated in Section 1.

4 Related Work

Most previous work on learning from conversational interactions has focused on the dialog sub-problems of response planning (e.g., Levin et al., 2000; Singh et al., 2002) and natural language generation (e.g., Lemon, 2011). We are not aware of previous work on inducing semantic parsers from conversations.

There has been significant work on supervised learning for inducing semantic parsers. Various techniques were applied to the problem including machine translation (Papineni et al., 1997; Ramaswamy and Kleindienst, 2000; Wong and Mooney, 2006; 2007; Matuszek et al., 2010), higher-order unification (Kwiatkowski et al., 2010), parsing (Ruifang and Mooney, 2006; Lu et al., 2008), inductive logic programming (Zelle and Mooney, 1996; Thompson and Mooney, 2003; Tang and Mooney, 2000), probabilistic push-down automata (He and Young, 2005; 2006) and ideas from support vector machines and string kernels (Kate and Mooney, 2006; Nguyen et al., 2006). The algorithms we develop in this paper build on previous work on supervised learning of CCG parsers (Zettlemoyer and Collins, 2005; 2007), as we describe in Section 5.3.

There is also work on learning to do semantic analysis with alternate forms of supervision. Clarke et al. (2010) and Liang et al. (2011) describe approaches for learning semantic parsers from questions paired with database answers, while Goldwasser et al. (2011) presents work on unsupervised learning. Our approach provides an alternative method of supervision that could complement these approaches. Additionally, there has been significant recent work on learning to do other, re-

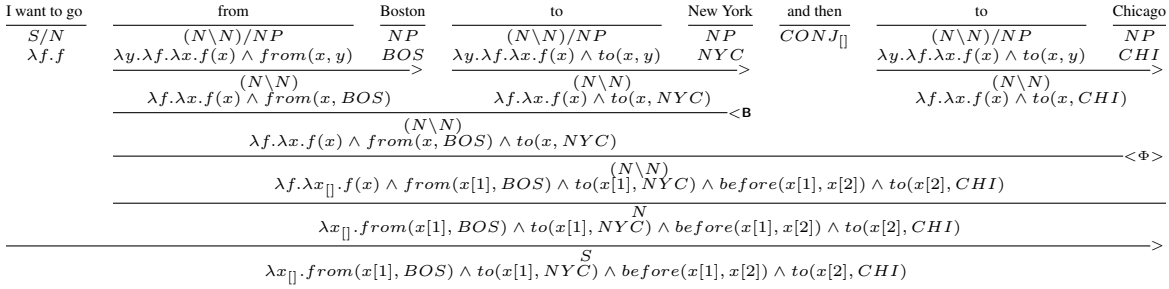


Figure 2: An example CCG parse. This parse shows the construction of a logical form with an array-typed variable $x_{[]}$ that specifies a list of flight legs, indexed by $x[1]$ and $x[2]$. The top-most parse steps introduce lexical items while the lower ones create new nonterminals according the CCG combinators ($>$, $<$, etc.), see Steedman (2000) for details.

lated, natural language semantic analysis tasks from context-dependent database queries (Miller et al., 1996; Zettlemoyer and Collins, 2009), grounded event streams (Chen et al., 2010; Liang et al., 2009), environment interactions (Branavan et al., 2009; 2010; Vogel and Jurafsky, 2010), and even unannotated text (Poon and Domingos, 2009; 2010).

Finally, the DARPA Communicator data (Walker et al., 2002) has been previously studied. Walker and Passonneau (2001) introduced a schema of speech acts for evaluation of the DARPA Communicator system performance. Georgila et al. (2009) extended this annotation schema to user utterances using an automatic process. Our speech acts extend this work to additionally include full meaning representations.

5 Mapping Sentences to Logical Form

We will use a weighted linear CCG grammar for semantic parsing, as briefly reviewed in this section.

5.1 Combinatory Categorial Grammars

Combinatory categorial grammars (CCGs) are a linguistically-motivated model for a wide range of language phenomena (Steedman, 1996; 2000). A CCG is defined by a lexicon and a set of combinators. The grammar defines a set of possible parse trees, where each tree includes syntactic and semantic information that can be used to construct logical forms for sentences.

The lexicon contains entries that define categories for words or phrases. For example, the second lexical entry in the parse in Figure 2 is:

$$\text{from} \;:=\; (N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge from(x, y)$$

Each category includes both syntactic and semantic information. For example, the phrase "from" is assigned the category with syntax $(N\backslash N)/NP$ and semantics $\lambda y.\lambda f.\lambda x.f(x) \wedge from(x, y)$. The outermost syntactic forward slash specifies that the entry must first be combined with an NP to the right (the departure city), while the inner back slash specifies that it will later modify a noun N to the left (to add a constraint to a set of flights). The lambda-calculus semantic expression is designed to build the appropriate meaning representation at each of these steps, as seen in the parse in Figure 2.

In general, we make use of typed lambda calculus to represent meaning (Carpenter, 1997), both in the lexicon and in intermediate parse tree nodes. We also introduce an extension for modeling array-typed variables to represent lists of individual entries. These constructions are used, for example, to model sentences describing a sequence of segments while specifying flight preferences.

Figure 2 shows how a CCG parse builds a logical form for a complete sentence with an array-typed variable. Each intermediate node in the tree is constructed with one of a small set of CCG combinator rules, see the explanation from Steedman (1996; 2000). We make use of the standard application, composition and coordination combinators, as well as type-shifting rules introduced by Zettlemoyer and Collins (2007) to model spontaneous, unedited text.

5.2 Weighted Linear CCGs

A weighted linear CCG (Clark and Curran, 2007) provides a ranking on the space of possible parses under the grammar, which can be used to select the best logical form for a sentence. This type of model is closely related to several other approaches (Ratnaparkhi et al., 1994; Johnson et al., 1999;

Lafferty et al., 2001; Collins, 2004; Taskar et al., 2004). Let x be a sentence, y be a CCG parse, and GEN$(x; \Lambda)$ be the set of all possible CCG parses for x given the lexicon Λ. Define $\phi(x, y) \in \mathbb{R}^d$ to be a d-dimensional *feature–vector* representation and $\theta \in \mathbb{R}^d$ to be a parameter vector. The optimal parse for sentence x is

$$y^*(x) = \arg \max_{y \in \text{GEN}(x;\Lambda)} \theta \cdot \phi(x, y)$$

and the final output logical form z is the lambda-calculus expression at the root of $y^*(x)$.

We compute $y^*(x)$ with a CKY-style chart parsing algorithm. Since each chart entry contains a full lambda-calculus meaning expression, we use N-best pruning to control the number of options we consider at each span. Learning a model of this form involves learning the parameters θ and the lexicon Λ. We will show that this is possible from conversational logs that do not contain any explicit labeling of the logical forms for user utterances.

5.3 Supervised learning with GENLEX

Previous work on lexical induction, including the GENLEX approach which we briefly review here, has required labeled logical meaning representations. In Section 7, we will introduce a new way of using GENLEX to learn from unannotated conversation logs.

The supervised CCG learning algorithms of Zettlemoyer and Collins (2005; 2007) induce a weighted CCG from training examples (x_i, z_i) for $i = 1 \ldots n$, where x_i is a sentence and z_i is the corresponding lambda-calculus meaning representation. The output from the algorithm is a pair (θ, Λ) containing the learned parameters and CCG lexicon.

They defined the function GENLEX(x, z) to map a sentence x and its meaning z to a large set of potential lexical entries. These entries are generated with rules that consider the logical form z and guess potential CCG categories. For example, the rule $p \rightarrow (N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge p(x, y)$ introduces categories commonly used to model certain types of prepositions. This rule would, for example, introduce the category $(N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x, y)$ for any logical form z that contains the constant to. GENLEX uses a small set of rules of this type to generate categories that are paired with all possible substrings in x to form an overly general lexicon. The complete learning algorithm then simultaneously selects a small subset of all entries generated by GENLEX and estimates parameter values θ. Zettlemoyer and Collins (2005) present a more detailed explanation.

Example Conversation:

(1) SYSTEM: how can i help you? (OPEN_TASK)

(2) USER: i would like to fly from boston arriving to new york and back from new york to boston

(3) SYSTEM: leaving boston (CONFIRM:$from(fl1, BOS)$) on what date? (ASK:$\lambda x.departdate(fl1, x)$)

(4) USER: september twenty four

(5) SYSTEM: leaving boston (CONFIRM:$from(fl1, BOS)$) on september twenty four (CONFIRM:$departdate(fl1, sept\_24)$) to new york (CONFIRM:$to(fl1, NYC)$) is new york your final destination? (ASK:$\neg\exists x.fr(x, NYC)$)

(6) USER: yes i want to return to boston

(7) SYSTEM: sorry, i didn't understand. (APOLOGY) where are you flying to from new york? (ASK:$\lambda x.fr(fl2, NYC) \wedge to(fl2, x)$)

[conversation ends]

Candidate Logical Expressions for Utterance #2:

(a) $\lambda x.to(x, BOS) \wedge from(x, NYC)$

(b) $\lambda x.from(x, BOS) \wedge to(x, NYC)$

(c) $\lambda x.to(x, BOS) \wedge to(x, NYC)$

(d) $\lambda x[].from(x[1], BOS) \wedge to(x[1], NYC)$
$\wedge before(x[1], x[2]) \wedge return(x[2])$
$\wedge from(x[2], NYC) \wedge to(x[2], BOS))$

(e) $\lambda x[].from(x[1], BOS) \wedge to(x[1], NYC)$
$\wedge before(x[1], x[2]) \wedge return(x[2])$
$\wedge from(x[2], BOS) \wedge to(x[2], NYC)$

Figure 3: Conversation reflecting an interaction as seen in the DARPA Communicator travel-planning dialogs.

6 Measuring Loss

In Section 7, we will present a loss-sensitive learning algorithm that models the meaning of user utterances as latent variables to be estimated from conversational interactions.

We first introduce a loss function to measure the quality of potential meaning representations. This loss function $\mathcal{L}(z, j, \mathcal{C}) \in \mathbb{R}$ indicates how well a logical expression z represents the meaning of the j-th user utterance in conversation $\mathcal{C}$. For example,

consider the first user utterance ($j = 2$) in Figure 3, which is a request for a return trip from Boston to New York. We would like to assign the lowest loss to the meaning representation (d) in Figure 3 that correctly encodes all of the stated constraints.

We make use of a loss function with two parts: $\mathcal{L}(z, j, \mathcal{C}) = \mathcal{L}_c(z, j, \mathcal{C}) + \mathcal{L}_d(z)$. The conversation loss $\mathcal{L}_c$ (defined in Section 6.1) measures how well the candidate meaning representation fits the conversation, for example incorporating information recovered through conversational remediations as motivated in Section 1. The domain loss $\mathcal{L}_d$ (described in Section 6.2) measures how well a logical form z matches domain expectations, such as the fact that flights can only have a single origin. These functions guide the types of meaning representations we expect to see, but in many cases will fail to specify a unique best option, for example in conversations where the user prematurely terminates the interaction. In Section 7, we will present a complete, loss-driven learning algorithm that is robust to these types of ambiguities while inducing a weighted CCG parser from conversations.

6.1 Conversation Loss

We will use a conversation loss function $\mathcal{L}_c(z, j, \mathcal{C})$ that provides a rough indication of how well the logical expression z represents a potential meaning for the user utterance at position j in $\mathcal{C}$. For example, the first user utterance ($j = 2$) in Figure 3 is a request for a return trip from Boston to New York where the user has explicitly mentioned both legs. The figure also shows five options (a-e) for the logical form z. We want to assign the lowest loss to option (d), which includes all of the stated constraints.

The loss is computed in four steps for a user utterance x at position j by (1) selecting a subset of system utterances in the conversation $\mathcal{C}$, (2) extracting and computing loss for semantic content from selected system utterances, (3) aligning the subexpressions in z to the extracted semantic content, and (4) computing the minimal loss value from the best alignment. In Figure 3, the loss for the candidate logical forms is computed by considering the segment of system utterances up until the conversation end. Within this segment, the matching for expression (d) involves mapping the origin and departure constraints for the first leg (Boston - New York) onto the earlier system confirmations while also aligning the ones for the second leg to system utterances later in the selected portion of the conversation. Finally, the overall score depends on the quality of the alignment, for example how many of the constraints match to confirmations. This section presents the full approach.

Segmentation For a user utterance at position j, we select all system utterances from $j - 1$ until the system believes it has completed the current subtask, as indicated by a reset action or final offer. We call this selected segment $\bar{\mathcal{C}}$. In Figure 3, $\bar{\mathcal{C}}$ ends with a reset, but in a successful interaction it would have ended with the offer of a specific flight.

Extracting Properties A property is a predicate-entity-value triplet, where the entity can be a variable from z or a conversational object. For example, $\langle from, fl, BOS \rangle$ is a property where fl is a object from $\bar{\mathcal{C}}$ and $\langle from, x, BOS \rangle$ is a property from $z = \lambda x.from(x, BOS)$. We define $P_{\bar{\mathcal{C}}}$ to be the set of properties from logical forms for system utterances in $\bar{\mathcal{C}}$. Similarly, we define P_z to be the set of properties in z.

Scoring System Properties For each system property $p \in P_{\bar{\mathcal{C}}}$ we compute its position value $pos(p)$, which is a normalized weighted average over all the positions where it appears in a logical form. For each mention the weight is obtained from its speech act. For example, properties that are explicitly confirmed contribute more to the average than those that were merely offered to the user in a select statement.

We use $pos(p)$ to compute a loss $loss(p)$ for each property $p \in P_{\bar{\mathcal{C}}}$. We first define $P^e_{\bar{\mathcal{C}}}$ to be all properties in $P_{\bar{\mathcal{C}}}$ with entity e. For entity e and position d, we define the entity-normalization function:

$$n_e(d) = \frac{d - \min_{p \in P^e_{\bar{\mathcal{C}}}} pos(p)}{\max_{p \in P^e_{\bar{\mathcal{C}}}} pos(p) - \min_{p \in P^e_{\bar{\mathcal{C}}}} pos(p)} \ .$$

For a given property $p \in P_{\bar{\mathcal{C}}}$ with an entity e we compute the loss value:

$$loss(p) = n_e^{-1}(1 - n_e(pos(p))) - 1 \ .$$

Where n_e^{-1} is the inverse of n_e. This loss value is designed to, first, provide less loss for later properties so that it, for example, favors the last property in a series of statements that finally resolves a confusion

in the conversation. Second, the loss value is lower for objects mentioned closer to the user utterance x, thereby preferring objects discussed sooner.

Matching Properties An alignment $\mathcal{A}$ maps variables in z to conversational objects in $\bar{\mathcal{C}}$, for example the flight legs $fl1$ and $fl2$ being discussed in Figure 3. We will use alignments to match properties of z and $\bar{\mathcal{C}}$. To do this we extend the alignment function $\mathcal{A}$ to apply to properties, for example $\mathcal{A}(\langle from, x, BOS\rangle) = \langle from, \mathcal{A}(x), BOS\rangle$.

Scoring Alignments Finally, we compute the conversation loss $\mathcal{L}_c(z, j, \mathcal{C})$ as follows:

$$\mathcal{L}_c(z, j, \mathcal{C}) = \min_{\mathcal{A}} \sum_{p_u \in P_z} \sum_{p_s \in P_{\bar{\mathcal{C}}}} s(\mathcal{A}(p_u), p_s) \quad .$$

The function $s(\mathcal{A}(p_u), p_s) \in \mathbb{R}$ computes the compatibility of the two input properties. It is zero if $\mathcal{A}(p_u) \neq p_s$. Otherwise, it returns $loss(p_s)$.

We approximate the $\min$ computation in $\mathcal{L}_c$ over alignments $\mathcal{A}$ as follows. For a logical form z at position j, we align the outer-most variable to the conversational object in $\bar{\mathcal{C}}$ that is being discussed at j. The remaining variables are aligned greedily to minimize the loss, by selecting a single conversational object for each in turn.

Finally, for each aligned variable, we increase the loss by one for each unmatched property from P_z. This increases the loss of logical forms that include spurious information. However, since a conversation might stop prematurely and therefore won't discuss the entire user request, we only increase the loss for variables that are already aligned. For this purpose, we define an aligned variable to be one that has at least one property matched successfully.

6.2 Domain Loss

We also make use of a domain loss function $\mathcal{L}_d(z) \in \mathbb{R}$. The function takes a logical form z and returns the number of violations there are in z to a set of constraints on logical forms that occur commonly in the dialog domain. For example, in a travel domain, a violation might occur if a flight leg has two different destination cities. The set of possible violations must be specified for each dialog system, but can often be compiled from existing resources, such as a database of valid flight ticketing options.

In our experiments, we will use a set of eight simple constraints to check for violations in flight itineraries, which can have multiple legs. These include, for example, checking that the legs have unique origins and destinations that match across the entire itinerary. For example, in Figure 3 the logical forms (a), (b) and (d) will have no violations; they describe valid flights. Example (c) has a single violation: a flight has two origins. Example (e) violates a more complex constraint: the second flight's origin is different from the first flight's destination.

Inputs: Training set $\{(j_i, \mathcal{C}_i) : i = 1 \ldots n\}$ where each example includes the index j_i of a sentence x_i in the conversation $\mathcal{C}_i$. Initial lexicon Λ_0. Number of iterations T. Margin γ. Beam size k for lexicon generation. Loss function $\mathcal{L}(x, j, \mathcal{C})$, as described in Section 6.

Definitions: $GENLEX(x, \mathcal{C})$ takes as input a sentence and a conversation and returns a set of lexical items as described in Section 7. GEN$(x; \Lambda)$ is the set of all possible CCG parses for x given the lexicon Λ. $LF(y)$ returns the logical form z at the root of the parse tree y. Let $\Phi_i(y)$ be shorthand for the feature function $\Phi(x_i, y)$ defined in Section 5. Define $LEX(y)$ to be the set of lexical entries used in parse y. Finally, let $MIN\mathcal{L}_i(Y)$ be $\{y | \forall y' \in Y, \mathcal{L}(LF(y), j_i, \mathcal{C}_i) \leq \mathcal{L}(LF(y'), j_i, \mathcal{C}_i)\}$, the set of minimal loss parses in Y.

Algorithm:

$\theta = \bar{0}$, $\Lambda = \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$:

Step 1: (Lexical generation)

a. Set $\lambda = \Lambda \cup GENLEX(x_i, \mathcal{C}_i)$
b. Let Y be the k highest scoring parses of x_i using λ
c. Select new lexical entries from the lowest loss parses $\lambda_i = \bigcup_{y \in MIN\mathcal{L}_i(Y)} \{l | l \in LEX(y)\}$
d. Set lexicon to $\Lambda = \Lambda \cup \lambda_i$

Step 2: (Update parameters)

a. Define $G_i = MIN\mathcal{L}_i(GEN(x_i, \Lambda, \theta))$ and $\mathcal{L}_{min}$ to be the minimal loss
b. Set $B_i = GEN(x_i, \Lambda, \theta) - G_i$
c. Set the relative loss function: $\Delta_i(y) = \mathcal{L}(y, \mathcal{C}_i) - \mathcal{L}_{min}$
d. Construct sets of margin violating good and bad parses:
$R_i = \{r | r \in G_i \wedge \exists y' \in B_i \; s.t. \; \langle \theta, \Phi_i(r) - \Phi_i(y') \rangle < \gamma \Delta_i(r)\}$
$E_i = \{e | e \in B_i \wedge \exists y' \in G_i \; s.t. \; \langle \theta, \Phi_i(y') - \Phi_i(e) \rangle < \gamma \Delta_i(e)\}$
e. Apply the additive update:
$\theta = \theta + \sum_{r \in R_i} \frac{1}{|R_i|} \Phi_i(r) - \sum_{e \in E_i} \frac{1}{|E_i|} \Phi_i(e)$

Output: Parameters θ and lexicon Λ

Figure 4: The learning algorithm.

7 Learning

Figure 4 presents the complete learning algorithm. We assume access to training examples, $\{(j_i, \mathcal{C}_i) : i = 1, \ldots, n\}$, where each example includes the in-

dex j_i of a sentence x_i in the conversation $\mathcal{C}_i$. The algorithm learns a weighted CCG parser, described in Section 5, including both a lexicon Λ and parameters θ. The approach is online, considering each example in turn and performing two steps: (1) expanding the lexicon and (2) updating the parameters.

Step 1: Lexical Induction We introduce new lexical items by selecting candidates from the function $GENLEX$, following previous work (Zettlemoyer and Collins, 2005; 2007) as reviewed in Section 5.3. However, we face the new challenge that there is no labeled logical-form meaning z. Instead, let $Z_{\bar{\mathcal{C}}}$ be set of all logical forms that appear in system utterances in the relevant conversation segment $\bar{\mathcal{C}}$. We will now define the conversational lexicon set:

$$GENLEX(x, \bar{\mathcal{C}}) = \bigcup_{z \in Z_{\bar{\mathcal{C}}}} GENLEX(x, z)$$

where we use logical forms from system utterances to guess possible CCG categories for analyzing the user utterance. This approach will overgeneralize, when the system talks about things that are unrelated to what the user said, and will also often be incomplete, for example when the system does not repeat parts of the original content. However, it provides a way of guessing lexical items that can be combined with previously learned ones, which can fill in any gaps and help select the best analysis.

Step 1(a) in Figure 4 uses $GENLEX$ to temporarily create a large set of potential categories based on the conversation. Steps (b-d) select a small subset of these entries to add to the current lexicon Λ: we find the k-best parses under the model, rerank them according to loss, find the lexical items used in the best trees, and add them to Λ. This approach favors lexical items that are used in high-scoring but low-loss analyses, as computed given the current model.

Step 2: Parameter Updates Given the loss function $\mathcal{L}(x, i, \mathcal{C})$, we use a variant of a loss-sensitive perceptron to update the parameters (Singh-Miller and Collins, 2007). In Steps (a-c), for the current example i, we compute the relative loss function Δ_i that scales with the loss achieved by the best and worst possible parses under the model. In contrast to previous work, we do not only compute the loss over a fixed n-best list of possible outputs, but instead use the current model score to recompute the options at each update. Then, Steps (d-e) find the set R_i of least loss analyses and E_i of higher-loss candidates whose models scores are not separated by at least $\gamma\Delta_i$, where γ is a margin scale constant. The final update (Step f) is additive and increases the parameters for features indicative of the analyses with less loss while down weighting those for parses that were not sufficiently separated.

Discussion This algorithm uses the conversation to drive learning in two ways: it guides the lexical items that are proposed while also providing the conversational feedback that defines the loss used to update the parameters. The resulting approach is, at every step, using information about how the conversation progressed after a user utterance to reconstruct the meaning of the original statement.

8 Data Sets

For evaluation, we used conversation logs from the Lucent and BBN dialog systems in the DARPA Communicator corpus (Walker et al., 2002). We selected these systems since they provide significant opportunities for learning. They asked relatively open ended questions, allowing for more complex user responses, while also using a number of simple remediating strategies to recover from misunderstandings. The original conversational logs included unannotated transcripts of system and user utterances. Inspired by the speech act labeling approach of Walker and Passonneau (2001), we wrote a set of scripts to label the speech acts and logical forms for system statements. This could be done with high accuracy since the original text was generated with templates. These labels represent what the system explicitly said and do not require complex, potentially error-prone annotation of the full state of the original dialog system. The set of speech acts includes confirmations, information requests, selects, offers, instructions, and a miscellaneous category.

The data sets include a total of 376 conversations, divided into training and testing sets. Table 1 provides details about the training and testing sets, as well as general data set statistics. We developed our system using 4-fold cross validation on the training sets. Although there are approximately 12,000 user

	Lucent	BBN
# Conversations	214	162
Total # of utterances	11,974	12,579
Avg. utterances per conversation	55.95	77.65
Avg. tokens per user utterance	3.24	2.39
Total # of training utterances	208	67
Total # of testing utterances	96	67
Avg. tokens per selected utterance	11.72	9.53

Table 1: Data set statistics for Lucent and BBN systems.

utterances in the data sets, the vast majority are simple, short phrases (such as "yes" or "no") which are not useful for learning a semantic parser. We select user utterances with a small set of heuristics, including a threshold (6 for Lucent, 4 for BBN) on the number of words and requiring that at least one noun phrase is present from our initial lexicon. This approach was manually developed to perform well on the training sets, but is not perfect and does introduce a small amount of noise into the data.

9 Experimental Setup

This section describes our experimental setup and comparisons. We follow the setup of Zettlemoyer and Collins (2007) where possible, including feature design, initialization of the semantic parser, and evaluation metrics, as reviewed below.

Features and Parser The features include indicators for lexical item use, properties of the logical form that is being constructed, and indicators for parsing operators used to build the tree. The parser attempts to boost recall with a two-pass strategy that allows for word skipping if the initial parse fails.

Initialization and Parameters We use an initial lexicon that includes a list of domain-specific noun phrases, such as city and airport names, and a list of domain-independent categories for closed-class words such as "the" and "and". We also used a time and number parser to expand this lexicon for each input sentence with the BIU Number Normalizer.[1] The learning parameters were tuned using the development sets: the margin constant γ is set to 0.5, we use 6 iterations and take the top 30 parses for lexical generation (step 1, figure 4). The parser used for parameter update (step 2, figure 4) has a beam of 250. The parameter vector is initialized to $\bar{0}$.

[1] http://www.cs.biu.ac.il/~nlp/downloads/

Evaluation Metrics For evaluation, we measure performance against gold standard labels. We report both the number of exact matches, fully correct logical forms, and a partial-credit number. We measure partial-credit accuracy by mapping logical forms to attribute-value pairs (for example, the expression $from(x, LA)$ will be mapped to $from = LA$) and report precision and recall on attribute sets. This more lenient measure does not test the overall structure of the logical expression, only its components.

Systems We compare performance with the following systems:

Full Supervision: We measured how a fully supervised approach would perform on our data by hand-labeling the training data and using a 0-1 loss function that tests if the output logical form matches the labeled one. For lexicon generation, the labels were used instead of the conversation.

No Conversation Baseline: We also report results for a no conversation baseline. This baseline system is constructed by making two modifications to the full approach. We remove the conversation loss function and apply the GENLEX templates to every possible logical constant, instead of only those in the conversation. This baseline allows us to measure the importance of having access to the conversations by completely ignoring the context for each sentence.

Ablations: In addition to the baseline above, we also do ablation tests by turning off various individual components of the complete algorithm.

10 Results

Table 2 shows exact match results for the development sets, including different system configurations. We report mean results across four folds. To verify their contributions, we include results where we ablate the conversational loss and domain loss functions. Both are essential.

The test results are listed in Table 3. The full method significantly outperforms the baseline, indicating that we are making effective use of the conversational feedback, although we do not yet match the fully supervised result. The poor baseline performance is not surprising, given the difficulty of the task and lack of guidance when the conversations are removed. The partial-credit numbers also demonstrate an empirical trend that we observed; in many

Exact Match Metric	Lucent			BBN		
	Prec.	Rec.	F1	Prec.	Rec.	F1
Without conversational loss	0.35	0.34	0.35	0.66	0.54	0.59
Without domain loss	0.42	0.42	0.42	0.69	0.56	0.61
Our Approach	0.63	0.61	0.62	0.77	0.64	0.69
Supervised method	0.76	0.75	0.75	0.81	0.67	0.73

Table 2: Mean exact-match results for cross fold evaluation on the development sets.

Exact Match Metric	Lucent			BBN		
	Prec.	Rec.	F1	Prec.	Rec.	F1
No Conversations Baseline	0	0	0	0.16	0.15	0.15
Our Approach	0.58	0.55	0.56	0.85	0.75	0.79
Supervised method	0.7	0.68	0.69	0.87	0.78	0.82
Partial Credit Metric	**Lucent**			**BBN**		
	Prec.	Rec.	F1	Prec.	Rec.	F1
No Conversations Baseline	0.26	0.35	0.29	0.26	0.33	0.29
Our Approach	0.68	0.63	0.65	0.97	0.57	0.72
Supervised method	0.75	0.68	0.72	0.96	0.68	0.79

Table 3: Exact- and partial-match results on the test sets.

cases where we do not produce the correct logical form, the output is often close to correct, with only one or two missed flight constraints.

The difference between the two systems is evident. The BBN system presents a simpler approach to the dialog problem by creating a more constrained conversation. This is done by handling one flight at a time, in the case of flight planing, and posing simple and close ended questions to the user. Such an approach encourages the user to make simpler requests, with relatively few constraints in each request. In contrast, the Lucent system presents a less-constrained approach: interactions start with an open ended prompt and the conversations flow in a more natural, less constrained fashion. BBN's simplified approach makes it easier for learning, giving us superior performance when compared to the Lucent system, despite the smaller training set. This is true for both our approach and supervised learning.

We compared the logical forms recovered by the best conversational model to the labeled ones in the training set. Many of the errors came from cases where the dialog system never fully recovered from confusions in the conversation. For example, the Lucent system almost never understood user utterances that specified flight arrival times. Since it was unable to consistently recover and introduce this constraint, the user would often just recalculate and specify a departure time that would achieve the original goal. This type of failure provides no signal for our learning algorithm, whereas the fully supervised algorithm would use labeled logical forms to resolve the confusion. Interestingly, the test set had more sentences that suffered such failures than the development set, which contributed to the performance gap.

11 Discussion

We presented a loss-driven learning approach that induces the lexicon and parameters of a CCG parser for mapping sentences to logical forms. The loss was defined over the conversational context, without requiring annotation of user utterances meaning.

The overall approach assumes that, in aggregate, the conversations contain sufficient signal (remediations such as clarification, etc.) to learn effectively. In this paper, we satisfied this requirement by using logs from automated systems that deployed reasonably effective recovery strategies. An important area for future work is to consider how this learning can be best integrated into a complete dialog system. This would include designing remediation strategies that allow for the most effective learning and considering how similar techniques could be used simultaneously for other dialog subproblems.

Acknowledgments

The research was supported by funding from the DARPA Computer Science Study Group. Thanks to Dan Weld, Raphael Hoffmann, Jonathan Berant, Hoifung Poon and Mark Yatskar for their suggestions and comments. We also thank Shachar Mirkin for providing access to the BIU Normalizer.

References

Allen, J., M. Manshadi, M. Dzikovska, and M. Swift. 2007. Deep linguistic processing for spoken dialogue systems. In *Proceedings of the Workshop on Deep Linguistic Processing*.

Branavan, SRK, H. Chen, L.S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

Branavan, SRK, L.S. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: learning to map high-level instructions to commands. In *Proceedings of the Association for Computational Linguistics*.

Carpenter, B. 1997. *Type-Logical Semantics*. The MIT Press.

Chen, D.L., J. Kim, and R.J. Mooney. 2010. Training a multilingual sportscaster: using perceptual context to learn language. *Journal of Artificial Intelligence Research* 37(1):397–436.

Clark, S. and J.R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33(4):493–552.

Clarke, J., D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Collins, M. 2004. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In *New Developments in Parsing Technology*.

Georgila, K., O. Lemon, J. Henderson, and J.D. Moore. 2009. Automatic annotation of context and speech acts for dialogue corpora. *Natural Language Engineering* 15(03):315–353.

Goldwasser, D., R. Reichart, J. Clarke, and D. Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings. of the Association of Computational Linguistics*.

He, Y. and S. Young. 2005. Semantic processing using the hidden vector state model. *Computer Speech and Language* 19:85–106.

He, Y. and S. Young. 2006. Spoken language understanding using the hidden vector state model. *Speech Communication* 48(3-4).

Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the Association for Computational Linguistics*.

Kate, R.J. and R.J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the Association for Computational Linguistics*.

Kwiatkowski, T., L.S. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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

Lemon, O. 2011. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. *Computer Speech & Language* 25(2):210–221.

Levin, E., R. Pieraccini, and W. Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing* 8(1):11–23.

Liang, P., M.I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the Association for Computational Linguistics the International Joint Conference on Natural Language Processing*.

Liang, P., M.I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Association for Computational Linguistics*.

Litman, D., J. Moore, M.O. Dzikovska, and E. Farrow. 2009. Using Natural Language Processing to Analyze Tutorial Dialogue Corpora Across Domains Modalities. In *Proceeding of the Conference on Artificial Intelligence in Education*.

Lu, W., H.T. Ng, W.S. Lee, and L.S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Matuszek, C., D. Fox, and K. Koscher. 2010. Following directions using statistical machine translation. In *Proceeding of the international conference on Human-robot interaction*.

Miller, S., D. Stallard, R.J. Bobrow, and R.L. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the Association for Computational Linguistics*.

Nguyen, L., A. Shimazu, and X. Phan. 2006. Semantic parsing with structured SVM ensemble classification models. In *Proceedings of the joint conference*

of the International Committee on Computational Linguistics and the Association for Computational Linguistics.

Papineni, K.A., S. Roukos, and T.R. Ward. 1997. Feature-based language understanding. In *Proceedings of the European Conference on Speech Communication and Technology*.

Poon, H. and P. Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Poon, H. and P. Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of the Association for Computational Linguistics*.

Ramaswamy, G.N. and J. Kleindienst. 2000. Hierarchical feature-based translation for scalable natural language understanding. In *Proceedings of the International Conference on Spoken Language Processing*.

Ratnaparkhi, A., S. Roukos, and R.T. Ward. 1994. A maximum entropy model for parsing. In *Proceedings of the International Conference on Spoken Language Processing*.

Ruifang, G. and R.J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Porceedings of the Association for Computational Linguistics*.

Singh, S.P., D.J. Litman, M.J. Kearns, and M.A. Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research* 16(1):105–133.

Singh-Miller, N. and M. Collins. 2007. Trigger-based language modeling using a loss-sensitive perceptron algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Steedman, M. 1996. *Surface Structure and Interpretation*. The MIT Press.

Steedman, M. 2000. *The Syntactic Process*. The MIT Press.

Tang, L.R. and R.J. Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Taskar, B., D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Thompson, C.A. and R.J. Mooney. 2003. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research* 18:1–44.

Vogel, A. and D. Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the Association for Computational Linguistics*.

Walker, M. and R. Passonneau. 2001. DATE: a dialogue act tagging scheme for evaluation of spoken dialogue systems. In *Proceedings of the First International Conference on Human Language Technology Research*.

Walker, M., A. Rudnicky, R. Prasad, J. Aberdeen, E. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, et al. 2002. DARPA Communicator: Cross-system results for the 2001 evaluation. In *Proceedings of the International Conference on Spoken Language Processing*.

Wong, Y.W. and R.J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Association for Computational Linguistics*.

Wong, Y.W. and R.J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Association for Computational Linguistics*.

Young, S., M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language* 24(2):150–174.

Zelle, J.M. and R.J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, L.S. and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, L.S. and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zettlemoyer, L.S. and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Timeline Generation through Evolutionary Trans-Temporal Summarization

Rui Yan[†], Liang Kong[†] , Congrui Huang[†], Xiaojun Wan[‡], Xiaoming Li[♮], Yan Zhang[†*]
[†]School of Electronics Engineering and Computer Science, Peking University, China
[‡]Institute of Computer Science and Technology, Peking University, China
[♮]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China
{r.yan,kongliang,hcr,lxm}@pku.edu.cn,
wanxiaojun@icst.pku.edu.cn,zhy@cis.pku.edu.cn

Abstract

We investigate an important and challenging problem in summary generation, i.e., Evolutionary Trans-Temporal Summarization (ETTS), which generates news timelines from massive data on the Internet. ETTS greatly facilitates fast news browsing and knowledge comprehension, and hence is a necessity. Given the collection of time-stamped web documents related to the evolving news, ETTS aims to return news evolution along the timeline, consisting of individual but correlated summaries on each date. Existing summarization algorithms fail to utilize trans-temporal characteristics among these component summaries. We propose to model trans-temporal correlations among component summaries for timelines, using *inter-date* and *intra-date* sentence dependencies, and present a novel combination. We develop experimental systems to compare 5 rival algorithms on 6 instinctively different datasets which amount to 10251 documents. Evaluation results in ROUGE metrics indicate the effectiveness of the proposed approach based on trans-temporal information.

1 Introduction

Along with the rapid growth of the World Wide Web, document floods spread throughout the Internet. Given a large document collection related to a news subject (for example, *BP Oil Spill*), readers get lost in the sea of articles, feeling confused and powerless. General search engines can rank these news webpages by *relevance* to a user specified aspect, i.e., a query such as "*first relief effort for BP Oil Spill*", but search engines are not quite capable of ranking documents given the whole news subject without particular aspects. Faced with thousands of news documents, people usually have a myriad of interest aspects about the beginning, the development or the latest situation. However, traditional information retrieval techniques can only rank webpages according to their understanding of relevance, which is obviously insufficient (Jin et al., 2010).

Even if the ranked documents could be in a satisfying order to help users understand news evolution, readers prefer to monitor the evolutionary trajectories by simply browsing rather than navigate every document in the overwhelming collection. Summarization is an ideal solution to provide an abbreviated, informative reorganization for faster and better representation of news documents. Particularly, a timeline (see Table 1) can summarize evolutionary news as a series of *individual* but *correlated* component summaries (items in Table 1) and offer an option to understand the big picture of evolution.

With unique characteristics, summarizing timelines is significantly different from traditional summarization methods which are awkward in such scenarios. We first study a manual timeline of BP Oil Spill in Mexico Gulf in Table 1 from Reuters News[1] to understand why timelines generation is observably different from traditional summarization. No traditional method has considered to partition corpus into subsets by timestamps for trans-temporal correlations. However, we discover two unique trans-

*Corresponding author.

[1]http://www.reuters.com

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 433–443,
Edinburgh, Scotland, UK, July 27–31, 2011.

Table 1: Part of human generated timeline about BP Oil Spill in 2010 from Reuters News website.

April 22, 2010
The Deepwater Horizon rig, valued at more than $560 million, sinks and a five mile long (8 km) oil slick is seen.
April 25, 2010
The Coast Guard approves a plan to have remote underwater vehicles activate a blowout preventer and stop leak. Efforts to activate the blowout preventer fail.
April 28, 2010
The Coast Guard says the flow of oil is 5,000 barrels per day (bpd) (210,000 gallons/795,000 litres) – five times greater than first estimated. A controlled burn is held on the giant oil slick.
April 29, 2010
U.S. President Barack Obama pledges "every single available resource," including the U.S. military, to contain the spreading spill. Obama also says BP is responsible for the cleanup. Louisiana declares state of emergency due to the threat to the state's natural resources.
April 30, 2010
An Obama aide says no drilling will be allowed in new areas, as the president had recently proposed, until the cause of the Deepwater Horizon accident is known.

temporal characteristics of component summaries from the handcrafted timeline. **Individuality.** The component summaries are summarized *locally*: the component item on date t is constituted by sentences with timestamp t. **Correlativeness.** The component summaries are correlative across dates, based on the *global* collection. To the best of our knowledge, no traditional method has examined the relationships among these timeline items.

Although it is profitable, summarizing timeline faces with new challenges:

• The first challenge for timeline generation is to deliver important contents and avoid information overlaps among component summaries under the trans-temporal scenario based on global/local source collection. Component items are individual but not completely isolated due to the dynamic evolution.

• As we have *individuality* and *correlativeness* to evaluate the qualities of component summaries, both locally and globally, the second challenge is to formulate the combination task into a balanced optimization problem to generate the timelines which satisfy both standards with maximum utilities.

We introduce a novel approach for the web mining problem Evolutionary Trans-Temporal Summarization (ETTS). Taking a collection relevant to a news subject as input, the system automatically outputs a timeline with items of component summaries which represent evolutionary trajectories on specific dates. We classify sentence relationships as *inter-date* and *intra-date* dependencies. Particularly, the inter-date dependency calculation includes temporal decays to project sentences from all dates onto the same time horizon (Figure 1 (a)). Based on intra-/inter-date sentence dependencies, we then model affinity and diversity to compute the saliency score of each sentence and merge local and global rankings into one unified ranking framework. Finally we select top ranked sentences. We build an experimental system on 6 real datasets to verify the effectiveness of our methods compared with other 4 rivals.

2 Related Work

Multi-document summarization (MDS) aims to produce a summary delivering the majority of information content from a set of documents and has drawn much attention in recent years. Conferences such as ACL, SIGIR, EMNLP, etc., have advanced the technology and produced several experimental systems.

Generally speaking, MDS methods can be either extractive or abstractive summarization. Abstractive summarization (e.g. NewsBlaster[2]) usually needs information fusion, sentence compression and reformulation. We focus on extraction-based methods, which usually involve assigning saliency scores to some units (e.g. sentences, paragraphs) of the documents and extracting the units with highest scores.

To date, various extraction-based methods have been proposed for generic multi-document summarization. The centroid-based method MEAD (Radev et al., 2004) is an implementation of the centroid-based method that scores sentences based on features such as cluster centroids, position, and TF.IDF, etc. NeATS (Lin and Hovy, 2002) adds new features such as topic signature and term clustering to select important content, and use MMR (Goldstein et al., 1999) to remove redundancy.

Graph-based ranking methods have been proposed to rank sentences/passages based on "votes" or "recommendations" between each other. TextRank (Mihalcea and Tarau, 2005) and LexPageRank (Erkan and Radev, 2004) use algorithms similar to PageRank and HITS to compute sentence importance. Wan et al. have improved the graph-ranking

[2]http://www1.cs.columbia.edu/nlp/newsblaster/

algorithm by differentiating intra-document and inter-document links between sentences (2007b), and have proposed a manifold-ranking method to utilize sentence-to-sentence and sentence-to-topic relationships (Wan et al., 2007a).

ETTS seems to be related to a very recent task of "update summarization" started in DUC 2007 and continuing with TAC. However, update summarization only dealt with a single update and we make a novel contribution with multi-step evolutionary updates. Further related work includes similar timeline systems proposed by (Swan and Allan, 2000) using named entities, by (Allan et al., 2001) measured in *usefulness* and *novelty*, and by (Chieu and Lee, 2004) measured in *interest* and *burstiness*. We have proposed a timeline algorithm named "Evolutionary Timeline Summarization (ETS)" in (Yan et al., 2011b) but the refining process based on generated component summaries is time consuming. We aim to seek for more efficient summarizing approach.

To the best of our knowledge, neither update summarization nor traditional systems have considered the relationship among "component summaries", or have utilized trans-temporal properties. ETTS approach can also naturally and simultaneously take into account global/local summarization with biased information richness and information novelty, and combine both summarization in optimization.

3 Trans-temporal Summarization

We conduct trans-temporal summarization based on the global biased graph using *inter-date* dependency and local biased graph using *intra-date* dependency. Each graph is the complementary graph to the other.

3.1 Global Biased Summarization

The intuition for global biased summarization is that the selected summary should be correlative with sentences from neighboring dates, especially with those informative ones. To generate the component summary on date t, we project all sentences in the collection onto the time horizon of t to construct a global affinity graph, using temporal decaying kernels.

3.1.1 Temporal Proximity Based Projection

Clearly, a major technical challenge in ETTS is how to define the temporal biased projection function $\Gamma(\Delta t)$, where Δt is the distance between the pending date t and neighboring date t', i.e., $\Delta t = |t' - t|$. As in (Lv and Zhai, 2009), we present 5 representative kernel functions: Gaussian, Triangle, Cosine, Circle, and Window, shown in Figure 2. Different kernels lead to different projections.

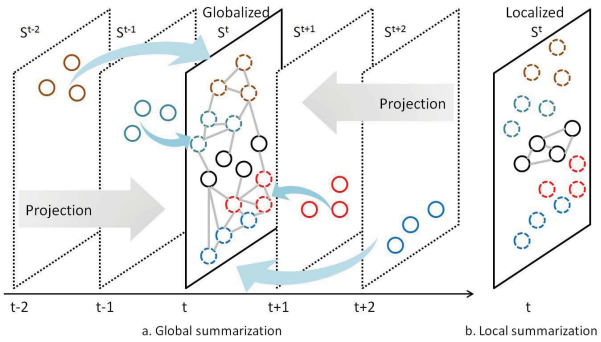


Figure 1: Construct global/local biased graphs. Solid circles denote intra-date sentences on the pending date t and dash ones represent inter-date sentences from other dates.

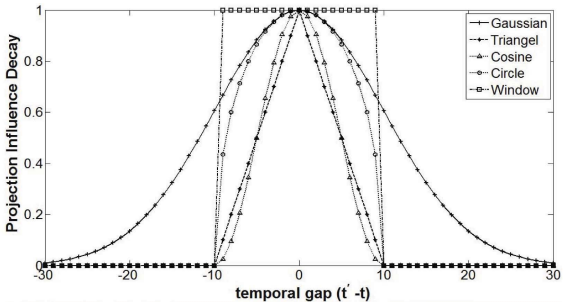


Figure 2: Proximity-based kernel functions, where σ=10.

1. Gaussian kernel

$$\Gamma(\Delta t) = \exp[\frac{-\Delta t^2}{2\sigma^2}]$$

2. Triangle kernel

$$\Gamma(\Delta t) = \begin{cases} 1 - \frac{\Delta t}{\sigma} & \text{if } \Delta t \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

3. Cosine (Hamming) kernel

$$\Gamma(\Delta t) = \begin{cases} \frac{1}{2}[1 + \cos(\frac{\Delta t \cdot \pi}{\sigma})] & \text{if } \Delta t \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

4. Circle kernel

$$\Gamma(\Delta t) = \begin{cases} \sqrt{1 - (\frac{\Delta t}{\sigma})^2} & \text{if } \Delta t \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

5. Window kernel

$$\Gamma(\Delta t) = \begin{cases} 1 & \text{if } \Delta t \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

All kernels have one parameter σ to tune, which controls the spread of kernel curves, i.e., it restricts the projection scope of each sentence. In general, the optimal setting of σ may vary according to the news set because sentences presumably would have wider semantic scope in certain news subjects, thus requiring a higher value of σ and vice versa.

3.1.2 Modeling Global Affinity

Given the sentence collection C partitioned by the timestamp set T, $C = \{C^1, C^2, \ldots, C^{|T|}\}$, we obtain $C^t = \{s_i^t | 1 \leq i \leq |C^t|\}$ where s_i is a sentence with the timestamp $t = t_{s_i}$. When we generate component summary on t, we project all sentences onto time horizon t. After projection, all sentences are weighted by their influence on t. We use an affinity matrix M^t with the entry of the inter-date transition probability on date t. The sum of each row equals to 1. Note that for the global biased matrix, we measure the affinity between local sentences from t and global sentences from other dates. Therefore, intra-date transition probability between sentences with the timestamp t is set to 0 for local summarization.

$M_{i,j}^t$ is the transition probability of s_i to s_j based on the perspective of date t, i.e., $p(s_i \rightarrow s_j|t)$:

$$p(s_i \rightarrow s_j|t) = \begin{cases} \frac{f(s_i \rightarrow s_j|t)}{\sum_{|C|} f(s_i \rightarrow s_k|t)} & \text{if } \sum f \neq 0 \\ 0 & \text{if } t_{s_i} = t_{s_j} = t \end{cases} \quad (1)$$

$f(s_i \rightarrow s_j|t)$ is defined as the temporal weighted cosine similarity between two sentences:

$$f(s_i \rightarrow s_j|t) = \sum_{w \in s_i \cap s_j} \pi(w, s_i|t) \cdot \pi(w, s_j|t) \quad (2)$$

where the weight π associated with term w is calculated with the temporal weighted *tf.isf* formula:

$$\pi(w, s|t) = \frac{\Gamma|t - t_s| \cdot tf(w, s)(1 + log(\frac{|C|}{N_w}))}{\sqrt{\sum_{|s|}(tf(w, s)(1 + log(\frac{|C|}{N_w})))^2}}. \quad (3)$$

where t_s is the timestamp of sentence s, and $tf(w, s)$ is the term frequency of w in s. t_s can be any date from T. $|C|$ is the sentences set size and N_w is the number of sentences containing term w.

We let $p(s_i \rightarrow s_i|t)$=0 to avoid self transition. Note that although $f(.)$ is a symmetric function, $p(s_i \rightarrow s_j|t)$ is usually not equal to $p(s_j \rightarrow s_i|t)$, depending on the degrees of nodes s_i and s_j.

Now we establish the affinity matrix $M_{i,j}^t$ and by using the general form of PageRank, we obtain:

$$\vec{\lambda} = \mu M^{-1} \vec{\lambda} + \frac{1 - \mu}{|C|} \vec{e} \quad (4)$$

where $\vec{\lambda}$ is the selective probability of all sentence nodes and $\vec{e}$ is a column vector with all elements equaling to 1. μ is the damping factor set as 0.85. Usually the convergence of the iteration algorithm is achieved when difference between the scores computed at two successive iterations for any sentences falls below a given threshold (0.0001 in this study).

3.1.3 Modeling Diversity

Diversity is to reflect both biased information richness and sentence novelty, which aims to reduce information redundancy. However, using standard PageRank of Equation (4) will not result in diversity. The aggregational effect of PageRank assigns high salient scores to closely connected node communities (Figure 3 (b)). A greedy vertex selection algorithm may achieve diversity by iteratively selecting the most prestigious vertex and then penalizing the vertices "covered" by the already selected ones, such as Maximum Marginal Relevance and its applications in Wan et al. (2007b; 2007a). Most recently diversity rank $DivRank$ is another solution to diversity penalization in (Mei et al., 2010).

We incorporate DivRank in our general ranking framework, which creates a dynamic M during each iteration, rather than a static one. After z times of iteration, the matrix M becomes:

$$M^{(z)} = \mu M^{(z-1)} \cdot \vec{\lambda}^{(z-1)} + \frac{1 - \mu}{|C|} \vec{e} \quad (5)$$

Equation (5) raises the probability for nodes with higher centrality and nodes already having high weights are likely to "absorb" the weights of its neighbors directly, and the weights of neighbors' neighbors indirectly. The process is to iteratively adjust matrix M according to $\vec{\lambda}$ and then to update $\vec{\lambda}$ according to the changed M. As iteration increases

there emerges a **rich-gets-richer** phenomenon (Figure 3 (c) and (d)). By incorporating DivRank, we obtain rank $r_i^\dagger$ and the global biased ranking score $\mathcal{G}_i$ for sentence s_i from date t to summarize C^t.

3.2 Local Biased Summarization

Naturally, the component summary for date t should be informative within C^t. Given the sentence collection $C^t = \{s_i^t | 1 \leq i \leq |C^t|\}$, we build an affinity matrix for Figure 1 (b), with the entry of intra-date transition probability calculated from standard cosine similarity. We incorporate DivRank within local summarization and we obtain the local biased rank and ranking score for s_i, denoted as $r_i^\ddagger$ and $\mathcal{L}_i$.

3.3 Optimization of Global/Local Combination

We do not directly add the global biased ranking score and local biased ranking score, as many previous works did (Wan et al., 2007b; Wan et al., 2007a), because even the same ranking score gap may indicate different rank gaps in two ranking lists.

Given subset C^t, let $R = \{r_i\}$(i = 1,...,$|C^t|$), r_i is the final ranking of s_i to estimate, optimize the following objective cost function $O(R)$,

$$\begin{aligned} O(R) =& \alpha \sum_{i=1}^{|C^t|} \mathcal{G}_i \| \frac{r_i}{\Psi_i} - \frac{r_i^\dagger}{\mathcal{G}_i} \|^2 \\ &+ \beta \sum_{i=1}^{|C^t|} \mathcal{L}_i \| \frac{r_i}{\Psi_i} - \frac{r_i^\ddagger}{\mathcal{L}_i} \|^2 \end{aligned} \tag{6}$$

where $\mathcal{G}_i$ is the global biased ranking score while $\mathcal{L}_i$ is the local biased ranking score. Ψ_i is expected to be the merged ranking score, namely sentence *importance*, which will be defined later. Among the two components in the objective function, the first component means that the refined rank should not deviate too much from the global biased rank. We use $\| \frac{r_i}{\Psi_i} - \frac{r_i^\dagger}{\mathcal{G}_i} \|^2$ instead of $\|r_i - r_i^\dagger\|^2$ in order to distinguish the differences between sentences from the same rank gap. The second component is similar by refining rank from local biased summarization.

Our goal is to find $R = R^*$ to minimize the cost function, i.e., $R^* = argmin\{O(R)\}$. R^* is the final rank merged by our algorithm. To minimize $O(R)$, we compute its first-order partial derivatives.

$$\frac{\partial O(R)}{\partial r_i} = \frac{2\alpha}{\Psi_i}(\frac{\mathcal{G}_i}{\Psi_i} r_i - r_i^\dagger) + \frac{2\beta}{\Psi_i}(\frac{\mathcal{L}_i}{\Psi_i} r_i - r_i^\ddagger) \tag{7}$$

Let $\frac{\partial O(R)}{\partial r_i} = 0$, we get

$$r_i^* = \frac{\alpha \Psi_i r_i^\dagger + \beta \Psi_i r_i^\ddagger}{\alpha \mathcal{G}_i + \beta \mathcal{L}_i} \tag{8}$$

Two special cases are that if (1) $\alpha = 0$, $\beta \neq 0$: we obtain $r_i = \Psi_i r_i^\ddagger / \mathcal{L}_i$, indicating we only use the local ranking score. (2) $\alpha \neq 0$, $\beta = 0$, indicating we ignore local ranking score and only consider global biased summarization using inter-date dependency.

There can be many ways to calculate the sentence importance Ψ_i. Here we define Ψ_i as the weighted combination of itself with ranking scores from global biased and local biased summarization:

$$\Psi_i^{(z)} = \frac{\alpha \mathcal{G}_i + \beta \mathcal{L}_i + \gamma \Psi_i^{(z-1)}}{\alpha + \beta + \gamma}. \tag{9}$$

To save one parameter we let $\alpha + \beta + \gamma = 1$. In the z-th iteration, $r_i^{(z)}$ is dependent on $\Psi_i^{(z-1)}$ and $\Psi_i^{(z)}$ is indirectly dependent on $r_i^{(z)}$ via $\Psi_i^{(z-1)}$. $\Psi_i^{(0)} = 0$. We iteratively approximate final Ψ_i for the ultimate rank list R^*. The expectation of stable Ψ_i is obtained when $\Psi_i^{(z)} = \Psi_i^{(z-1)}$. Final Ψ_i is expected to satisfy $\Psi_i = \alpha \mathcal{G}_i + \beta \mathcal{L}_i + \gamma \Psi_i$:

$$\Psi_i = \frac{\alpha \mathcal{G}_i + \beta \mathcal{L}_i}{1 - \gamma} = \frac{\alpha \mathcal{G}_i + \beta \mathcal{L}_i}{\alpha + \beta} \tag{10}$$

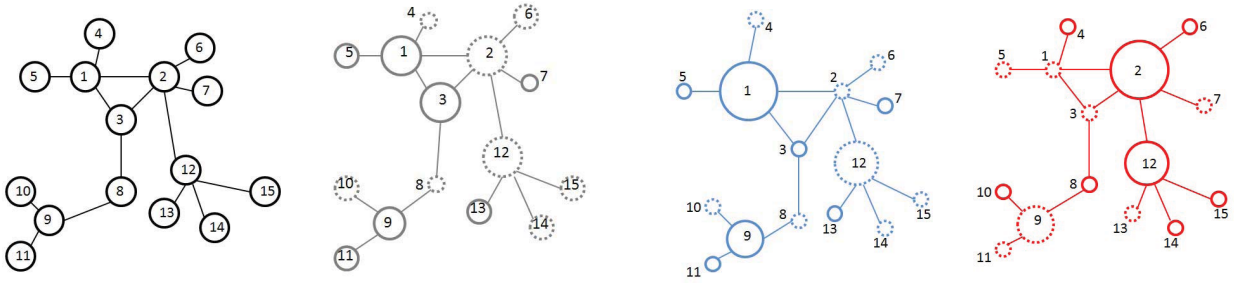
Final Ψ_i is dependent only on original global/local biased ranking scores. Equation (8) becomes more concise with no Ψ or γ: r^* is a weighted combination of global and local ranks by $\frac{\alpha}{\beta}$ ($\alpha \neq 0$, $\beta \neq 0$):

$$\begin{aligned} r_i^* &= \frac{\alpha}{\alpha + \beta} r_i^\dagger + \frac{\beta}{\alpha + \beta} r_i^\ddagger \\ &= \frac{1}{1 + \beta/\alpha} r_i^\dagger + \frac{1}{1 + \alpha/\beta} r_i^\ddagger \end{aligned} \tag{11}$$

4 Experiments and Evaluation

4.1 Datasets

There is no existing standard test set for ETTS methods. We randomly choose 6 news subjects with special coverage and handcrafted timelines by editors from 10 selected news websites: these 6 test sets consist of news datasets and golden standards to evaluate our proposed framework empirically, which amount to 10251 news articles. As shown in Table 2, three of the sources are in UK, one of them



(a) An illustrative network. (a) PageRank on t. (b) DivRank on t (c) DivRank on t'

Figure 3: An illustration of diverse ranking in a toy graph (a). Comparing (b) from general PageRank with (c),(d) from DivRank, we find a better diversity by selecting {1,9} in (c) rather than {1,3} in (b). Moreover, (c) and (d) reflect temporal biased processes on t {1,9} in (c) and t' {2,12} in (d).

is in China and the rest are in the US. We choose these sites because many of them provide timelines edited by professional editors, which serve as reference summaries. The news belongs to different categories of Rule of Interpretation (ROI) (Kumaran and Allan, 2004). More detailed statistics are in Table 3.

Table 2: News sources of 6 datasets

News Sources	Nation	News Sources	Nation
BBC	UK	Fox News	US
Xinhua	China	MSNBC	US
CNN	US	Guardian	UK
ABC	US	New York Times	US
Reuters	UK	Washington Post	US

Table 3: Detailed basic information of 6 datasets.

News Subjects	#size	#docs	#stamps	#RT	AL
1.Influenza A	115026	2557	331	5	83
2.Financial Crisis	176435	2894	427	2	118
3.BP Oil Spill	63021	1468	135	6	76
4.Haiti Earthquake	12073	247	83	2	32
5.Jackson Death	37819	925	168	3	64
6.Obama Presidency	79761	2160	349	5	92

size: the whole sentence counts; #stamps: the number of timestamps; Note **average size** of subsets is calculated as: avg.size=#size/#stamps; RT: reference timelines; AL: avg. length of RT measured in sentences.

4.2 Experimental System Setups

• **Preprocessing.** As ETTS faces with much larger corpus compared with traditional MDS, we apply further data preprocessing besides stemming and stop-word removal. We extract *text snippets* representing atomic "events" from all documents with a toolkit provided by Yan et al. (2010; 2011a), by which we attempt to assign more fine-grained and accurate timestamps for every sentence within the text snippets. After the snippet extraction procedure, we filter the corpora by discarding non-event texts.

• **Compression Rate and Date Selection.** After preprocessing, we obtain numerous snippets with fine-grained timestamps, and then decompose them into temporally tagged sentences as the global collection C. We partition C according to timestamps of sentences, i.e., $C = C^1 \cup C^2 \cup \cdots \cup C^{|T|}$. Each component summary is generated from its corresponding sub-collection. The sizes of component summaries are not necessarily equal, and moreover, not all dates may be represented, so date selection is also important. We apply a simple mechanism that users specify the overall compression rate ϕ, and we extract more sentences for important dates while fewer sentences for others. The *importance* of dates is measured by the *burstiness*, which indicates probable significant occurrences (Chieu and Lee, 2004). The compression rate on t_i is set as $\phi_i = \frac{|C^i|}{|C|}$.

4.3 Evaluation Metrics

The ROUGE measure is widely used for evaluation (Lin and Hovy, 2003): the DUC contests usually officially employ ROUGE for automatic summarization evaluation. In ROUGE evaluation, the summarization quality is measured by counting the number of overlapping units, such as N-gram, word sequences, and word pairs between the candidate timelines CT and the reference timelines RT. There are several kinds of ROUGE metrics, of which the most important one is ROUGE-N with 3 sub-metrics:

1 ROUGE-N-R is an N-gram recall metric:

$$\text{ROUGE-N-R} = \frac{\sum_{\text{I}\in\text{RT}} \sum_{\text{N-gram}\in\text{I}} \text{Count}_{\text{match}}(\text{N-gram})}{\sum_{\text{I}\in\text{RT}} \sum_{\text{N-gram}\in\text{I}} \text{Count (N-gram)}}$$

2 ROUGE-N-P is an N-gram precision metric:

$$\text{ROUGE-N-P} = \frac{\sum_{\text{I}\in\text{CT}} \sum_{\text{N-gram}\in\text{I}} \text{Count}_{\text{match}}(\text{N-gram})}{\sum_{\text{I}\in\text{CT}} \sum_{\text{N-gram}\in\text{I}} \text{Count (N-gram)}}$$

3 ROUGE-N-F is an N-gram F_1 metric:

$$\text{ROUGE-N-F} = \frac{2 \times \text{ROUGE-N-P} \times \text{ROUGE-N-R}}{\text{ROUGE-N-P} + \text{ROUGE-N-R}}$$

I denotes a timeline. N in these metrics stands for the length of N-gram and N-gram$\in$RT denotes the N-grams in reference timelines while N-gram$\in$CT denotes the N-grams in the candidate timeline. $\text{Count}_{\text{match}}$(N-gram) is the maximum number of N-gram in the candidate timeline and in the set of reference timelines. $\text{Count}_{\text{(N-gram)}}$ is the number of N-grams in reference timelines or candidate timelines.

According to (Lin and Hovy, 2003), among all sub-metrics, unigram-based ROUGE (ROUGE-1) has been shown to agree with human judgment most and bigram-based ROUGE (ROUGE-2) fits summarization well. We report three ROUGE F-measure scores: ROUGE-1, ROUGE-2, and ROUGE-W, where ROUGE-W is based on the weighted longest common subsequence. The weight W is set to be 1.2 in our experiments by ROUGE package (version 1.55). Intuitively, the higher the ROUGE scores, the similar the two summaries are.

4.4 Algorithms for Comparison

We implement the following widely used summarization algorithms as baseline systems. They are designed for traditional summarization without trans-temporal dimension. The first intuitive way to generate timelines by these methods is via a global summarization on collection C and then distribution of selected sentences to their source dates. The other one is via an equal summarization on all local sub-collections. For baselines, we average both intuitions as their performance scores. For fairness we conduct the same preprocessing for all baselines.

Random: The method selects sentences randomly for each document collection.

Centroid: The method applies MEAD algorithm (Radev et al., 2004) to extract sentences according to the following three parameters: centroid value, positional value, and first-sentence overlap.

GMDS: The graph-based MDS proposed by (Wan and Yang, 2008) first constructs a sentence connectivity graph based on cosine similarity and then selects important sentences based on the concept of eigenvector centrality.

Chieu: (Chieu and Lee, 2004) present a similar timeline system with different goals and frameworks, utilizing *interest* and *burstiness* ranking but neglecting trans-temporal news evolution.

ETTS: ETTS is an algorithm with optimized combination of global/local biased summarization.

RefTL: As we have used multiple human timelines as references, we not only provide ROUGE evaluations of the competing systems but also of the human timelines against each other, which provides a good indicator as to the upper bound ROUGE score that any system could achieve.

4.5 Overall Performance Comparison

We use a **cross validation** manner among 6 datasets, i.e., train parameters on one subject set and examine the performance on the others. After 6 training-testing processes, we take the average F-score performance in terms of ROUGE-1, ROUGE-2, and ROUGE-W on all sets. The overall results are shown in Figure 4 and details are listed in Tables 4$\sim$6.

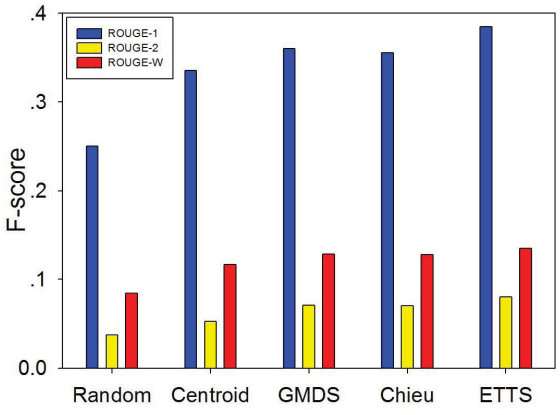


Figure 4: Overall performance on 6 datasets.

From the results, we have following observations:

• Random has the worst performance as expected.

• The results of Centroid are better than those of Random, mainly because the Centroid method takes

Table 4: Overall performance comparison on Influenza A (ROI* category: Science) and Financial Crisis (ROI category: Finance). α=0.4, kernel=Gaussian, σ=60.

	1. Influenza A			2. Financial Crisis		
Systems	R-1	R-2	R-W	R-1	R-2	R-W
RefTL	0.491	0.114	0.161	0.458	0.112	0.159
Random	0.257	0.039	0.081	0.230	0.030	0.071
Centroid	0.331	0.050	0.114	0.305	0.041	0.108
GMDS	0.364	0.062	0.130	0.327	0.054	0.110
Chieu	0.350	0.059	0.128	0.325	0.052	0.109
ETTS	**0.375**	**0.071**	**0.132**	**0.339**	**0.058**	**0.112**

Table 5: Overall performance comparison on BP Oil (ROI category: Accidents) and Haiti Quake (ROI category: Disasters). α=0.4, kernel=Gaussian, σ=30.

	3. BP Oil			4. Haiti Quake		
Systems	R-1	R-2	R-W	R-1	R-2	R-W
RefTL	0.517	0.135	0.183	0.528	0.139	0.187
Random	0.262	0.041	0.096	0.266	0.043	0.093
Centroid	0.369	0.062	0.128	0.362	0.060	0.129
GMDS	0.389	0.084	0.139	0.380	0.106	0.137
Chieu	0.384	0.083	0.139	0.383	0.110	0.138
ETTS	**0.441**	**0.107**	**0.158**	**0.436**	**0.111**	**0.145**

Table 6: Overall performance comparison on Jackson Death (ROI category: Legal Cases) and Obama Presidency (ROI category: Politics). α=0.4, kernel=Gaussian, σ=30.

	5. Jackson Death			6. Obama Presidency		
Systems	R-1	R-2	R-W	R-1	R-2	R-W
RefTL	0.482	0.113	0.161	0.495	0.115	0.163
Random	0.232	0.033	0.080	0.254	0.039	0.084
Centroid	0.320	0.051	0.109	0.325	0.053	0.111
GMDS	0.341	0.059	0.127	0.359	0.061	0.129
Chieu	0.344	0.059	0.128	0.346	0.060	0.125
ETTS	**0.358**	**0.061**	**0.130**	**0.369**	**0.074**	**0.133**

*ROI: news categorization defined by Linguistic Data Consortium.

into account positional value and first-sentence overlap, which facilitate main aspects summarization.

• The GMDS system outperforms centroid-based summarization methods. This is due to the fact that PageRank-based framework ranks the sentence using eigenvector centrality which implicitly accounts for information subsumption among all sentences.

Traditional MDS only consider sentence selection from either the global or the local scope, and hence bias occurs. Mis-selected sentences result in a low recall. Generally the performance of global priority intuition (i.e. only global summarization and then distribution to temporal subsets) is better than local priority methods (only local summarization). Probable bias is enlarged by searching for worthy sentence in single dates. However, precision drops due to excessive choice of global timeline-worthy sentences.

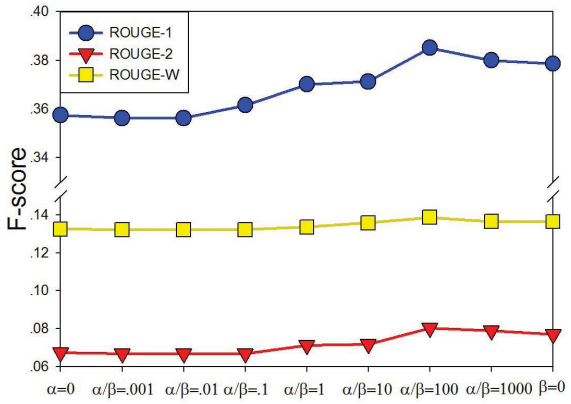


Figure 5: α/β: global/local combination.

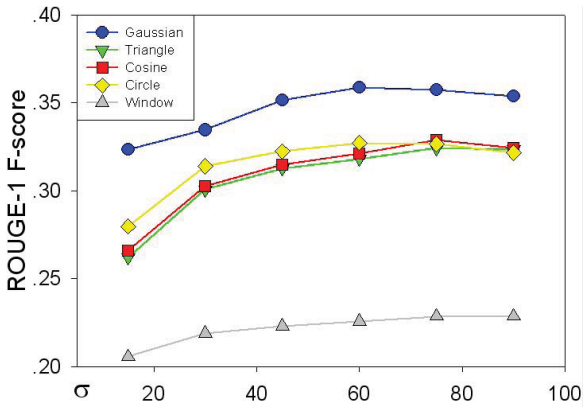


Figure 6: σ on long topics ($\geq$1 year).

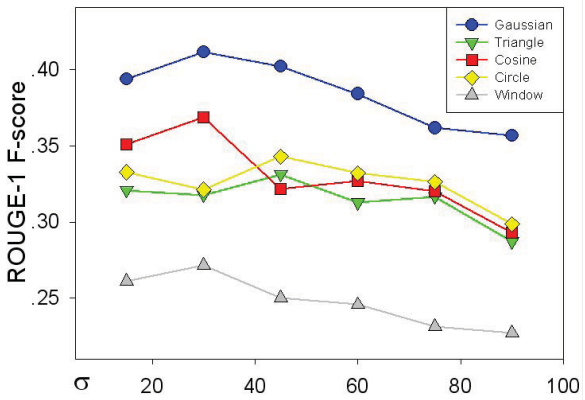


Figure 7: σ on short topics ($<$1 year).

• In general, the result of Chieu is better than Centroid but unexpectedly, worse than GMDS. The reason may be that Chieu does not capture sufficient timeline attributes. The "interest" modeled

in the algorithms actually performs flat clustering-based summarization which is proved to be less useful (Wang and Li, 2010). GMDS utilizes sentence linkage, and partly captures "correlativeness".

• ETTS under our proposed framework outperforms baselines, indicating that the properties we use for timeline generation are beneficial. We also add a direct comparison between ETTS and ETS (Yan et al., 2011b). We notice that both balanced algorithms achieve comparable performance (0.386 v.s. 0.412: a gap of 0.026 in terms of ROUGE-1), but ETTS is much faster than ETS. It is understandable that ETS refines timelines based on neighboring component summaries iteratively while for ETTS neighboring information is incorporated in temporal projection and hence there is no such procedure. Furthermore, ETS has 8 free parameters to tune while ETTS has only 2 parameters. In other words, ETTS is more simple to control.

• The performance on intensive focused news within short time range (|last timestamp−first timestamp |<1 year) is better than on long lasting news.

Having proved the effectiveness of our proposed methods, we carry the next move to identity how *global−local combination ratio* α/β and *projection kernels* take effects to enhance the quality of a summary in parameter tuning.

4.6 Parameter Tuning

Each time we tune one parameter while others are fixed. To identify how *global* and *local* biased summarization combine, we provide experiments on the performance of varying α/β in Figure 5. Results indicate that a balance between global and local biased summarization is essential for timeline generation because the performance is best when $\frac{\alpha}{\beta} \in [10, 100]$ and outperforms global and local summarization in isolation, i.e., when α=0 or $\beta = 0$ in Figure 5. Interestingly, we conclude an opposite observation compared with ETS. Different approaches might lead to different optimum of global/local combination.

Another key parameter σ measures the temporal projection influence from global collection to local collection and hence the size of neighboring sentence set. 6 datasets are classified into two groups. Subject 1, 2, 6 are grouped as long news with a time span of more than one year and the others are short news. The effect of σ varies on long news sets and short news sets. In Figure 6 σ is best around 60 and in Figure 7 it is best at about 20∼40, indicating long news has relatively wider semantic scope.

We then examine the effect of different projection kernels. Generally, Gaussian kernel outperforms others and window kernel is the worst, probably because Gaussian kernel provides the best smoothing effect with no arbitrary cutoffs. Window kernel fails to distinguish different weights of neighboring sets by temporal proximity, so its performance is as expected. Other 3 kernels are comparable.

4.7 Sample Output and Case Study

Sample output is presented in Table 7 and it shares major information similarity with the human timeline in Table 1. Besides, we notice that a dynamic ϕ_i is reasonable. Important burstiness is worthy of more attention. Fewer sentences are selected on the dates when nothing new occurs.

Interesting Findings. We notice that humans have biases to generate timelines for they have (1) preference on local occurrences and (2) different writing styles. For instance, news outlets from United States tend to summarize reactions by US government while UK websites tend to summarize British affairs. Some editors favor statistical reports while others prefer narrative style, and some timelines have detailed explanations while others are extremely concise with no more than two sentences for each entry. Our system-generated timelines have a large variance among all golden standards. Probably a new evaluation metric should be introduced to measure the quality of human generated timelines to mitigate the corresponding biases. A third interesting observation is that subjects have different volume patterns, e.g., $H1N1$ has a slow start and a bursty evolution and *BP Oil* has a bursty start and a quick decay. $Obama$ is different in nature because the report volume is temporally stable and scattered.

5 Conclusion

We present a novel solution for the important web mining problem, Evolutionary Trans-Temporal Summarization (ETTS), which generates trajectory timelines for news subjects from massive data. We formally formulate ETTS as a combination of *global* and *local* summarization, incorporating affinity and

Table 7: Selected part of timeline generated by ETTS for $BP\ Oil$.

April 20, 2010	**April 24, 2010**
s_1: An explosion on the Deepwater Horizon offshore oil drilling rig in the Gulf of Mexico, around 40 miles south east of Louisiana, causing several kills and injuries. s_2: The rig was drilling in about 5,000ft (1,525m) of water, pushing the boundaries of deepwater drilling technology. s_3: The rig is owned and operated by Transocean, a company hired by BP to carry out the drilling work. s_4: Deepwater Horizon oil rig fire leaves 11 missing.	s_1: Oil is found to be leaking from the well.
April 22, 2010	**April 26, 2010**
s_1: The US Coast Guard estimates that the rig is leaking oil at the rate of up to 8,000 barrels a day. s_2: The Deepwater Horizon sinks to the bottom of the Gulf after burning for 36 hours, raising concerns of a catastrophic oil spill. s_3: Deepwater Horizon rig sinks in 5,000ft of water.	s_1: BP's shares fall 2% amid fears that the cost of cleanup and legal claims will hit the London-based company hard. s_2: Roughly 15,000 gallons of dispersants and 21,000ft of containment boom are placed at the spill site.
April 23, 2010	**April 27, 2010**
s_1: The US coast guard suspends the search for missing workers, who are all presumed dead. s_2: The Coast Guard says it had no indication that oil was leaking from the well 5,000ft below the surface of the Gulf. s_3: Underwater robots try to shut valves on the blowout preventer to stop the leak, but BP abandons that failed effort two weeks later. s_4: The US Coast Guard estimates that the rig is leaking oil at the rate of up to 8,000 barrels a day. s_5: Deepwater Horizon clean-up workers fight to prevent disaster.	s_1: BP reports a rise in profits, due in large part to oil price increases, as shares rise again. s_2: The US departments of interior and homeland security announce plans for a joint investigation of the explosion and fire. s_3: Minerals Management Service (MMS) approves a plan for two relief wells. s_4: BP chairman Tony Hayward says the company will take full responsibility for the spill, paying for legitimate claims and cleanup cost.
	April 28, 2010
	s_1: The coast guard says the flow of oil is 5,000bpd, five times greater than first estimated, after a third leak is discovered. s_2: BP's attempts to repair a hydraulic leak on the blowout preventer valve are unsuccessful. s_3: BP reports that its first-quarter profits more than double to £3.65 billion following a rise in oil prices. s_4: Controlled burns begin on the giant oil slick.

diversity into a unified ranking framework. We implement a system under such framework for experiments on real web datasets to compare all approaches. Through our experiment we notice that the combination plays an important role in timeline generation, and global optimization weights slightly higher ($\alpha/\beta \in [10, 100]$), but auxiliary local information does help to enhance performance in ETTS.

Acknowledgments

This work was partially supported by NSFC with Grant No.61073082, 60933004, 70903008 and 61073081, and Xiaojun Wan was supported by NSFC with Grant No.60873155 and Beijing Nova Program (2008B03).

References

James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal summaries of new topics. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 10–18.

Hai Leong Chieu and Yoong Keok Lee. 2004. Query based event extraction along a timeline. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 425–432.

G. Erkan and D.R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP*, volume 4.

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. 1999. Summarizing text documents: sentence selection and evaluation metrics. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 121–128.

Xin Jin, Scott Spangler, Rui Ma, and Jiawei Han. 2010. Topic initiator detection on the world wide web. In *Proceedings of the 19th international conference on WWW'10*, pages 481–490.

Giridhar Kumaran and James Allan. 2004. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR'04*, pages 297–304.

Chin-Yew Lin and Eduard Hovy. 2002. From single to multi-document summarization: a prototype system and its evaluation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 457–464.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the Human Language Technology Conference of the NAACL'03*, pages 71–78.

Yuanhua Lv and ChengXiang Zhai. 2009. Positional language models for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 299–306.

Qiaozhu Mei, Jian Guo, and Dragomir Radev. 2010. Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD'10*, pages 1009–1018.

R. Mihalcea and P. Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP*, volume 5.

D.R. Radev, H. Jing, and M. Sty. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40(6):919–938.

Russell Swan and James Allan. 2000. Automatic generation of overview timelines. In *Proceedings of the 23rd annual international ACM SIGIR'00*, pages 49–56.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 299–306.

X. Wan, J. Yang, and J. Xiao. 2007a. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of IJCAI*, volume 7, pages 2903–2908.

X. Wan, J. Yang, and J. Xiao. 2007b. Single document summarization with document expansion. In *Proceedings of the 22nd AAAI'07*, pages 931–936.

Dingding Wang and Tao Li. 2010. Document update summarization using incremental hierarchical clustering. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 279–288.

Rui Yan, Yu Li, Yan Zhang, and Xiaoming Li. 2010. Event recognition from news webpages through latent ingredients extraction. In *Information Retrieval Technology - 6th Asia Information Retrieval Societies Conference, AIRS 2010*, pages 490–501.

Rui Yan, Liang Kong, Yu Li, Yan Zhang, and Xiaoming Li. 2011a. A fine-grained digestion of news webpages through event snippet extraction. In *Proceedings of the 20th international conference companion on world wide web*, WWW '11, pages 157–158.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011b. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *Proceedings of the 34th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '11.

Corpus-Guided Sentence Generation of Natural Images

Yezhou Yang † and **Ching Lik Teo** † and **Hal Daumé III** and **Yiannis Aloimonos**
University of Maryland Institute for Advanced Computer Studies
College Park, Maryland 20742, USA
{yzyang, cteo, hal, yiannis}@umiacs.umd.edu

Abstract

We propose a sentence generation strategy that describes images by predicting the most likely nouns, verbs, scenes and prepositions that make up the core sentence structure. The input are initial noisy estimates of the objects and scenes detected in the image using state of the art trained detectors. As predicting actions from still images directly is unreliable, we use a language model trained from the English Gigaword corpus to obtain their estimates; together with probabilities of co-located nouns, scenes and prepositions. We use these estimates as parameters on a HMM that models the sentence generation process, with hidden nodes as sentence components and image detections as the emissions. Experimental results show that our strategy of combining vision and language produces readable and descriptive sentences compared to naive strategies that use vision alone.

1 Introduction

What happens when you see a picture? The most natural thing would be to *describe* it using *words*: using speech or text. This description of an image is the output of an extremely complex process that involves: 1) perception in the Visual space, 2) grounding to World Knowledge in the Language Space and 3) speech/text production (see Fig. 1). Each of these components are challenging in their own right and are still considered open problems in the vision and linguistics fields. In this paper, we introduce a computational framework that attempts to integrate these components together. Our hypothesis is based on the assumption that natural images accurately reflect common everyday scenarios which are captured in language. For example, knowing that boats usually occur over `water` will enable us to constrain the possible scenes a boat can occur and exclude highly unlikely ones – `street, highway`. It also enables us to predict likely actions (Verbs) given the current object detections in the image: detecting a dog with a person will likely induce `walk` rather than `swim, jump, fly`. Key to our approach is the use of a large generic corpus such as the English Gigaword [Graff, 2003] as the *semantic grounding* to predict and correct the initial and often noisy visual detections of an image to produce a reasonable sentence that succinctly describes the image.

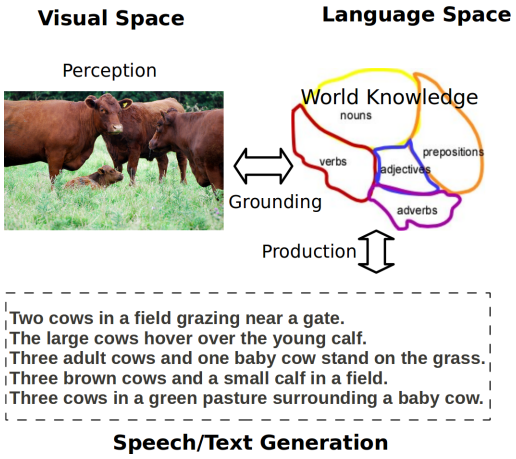


Figure 1: The processes involved for describing a scene.

In order to get an idea of the difficulty of this task, it is important to first define what makes up

† indicates equal contribution.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 444–454,
Edinburgh, Scotland, UK, July 27–31, 2011.

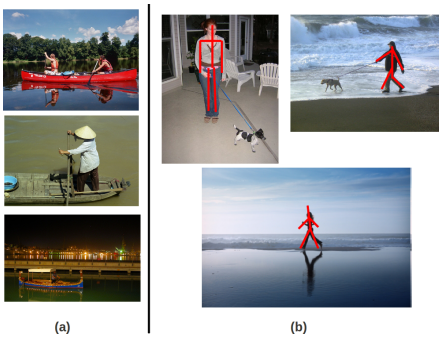


Figure 2: Illustration of various perceptual challenges for sentence generation for images. (a) Different images with semantically the same content. (b) Pose relates ambiguously to actions in real images. See text for details.

a description of an image. Based on our observations of annotated image data (see Fig. 4), a descriptive sentence for an image must contain at minimum: 1) the important *objects* (Nouns) that participate in the image, 2) Some description of the *actions* (Verbs) associated with these objects, 3) the *scene* where this image was taken and 4) the *preposition* that relates the objects to the scene. That is, a quadruplet of $\mathcal{T} = \{n, v, s, p\}$ (Noun-Verb-Scene-Preposition) that represents the core sentence structure. Generating a sentence from this quadruplet is obviously a simplification from state of the art generation work, but as we will show in the experimental results (sec. 4), it is sufficient to describe images. The key challenge is that detecting objects, actions and scenes *directly* from images is often noisy and unreliable. We illustrate this using example images from the Pascal-Visual Object Classes (VOC) 2008 challenge [Everingham et al., 2008]. First, Fig. 2(a) shows the *variability* of images in their raw image representations: pixels, edges and local features. This makes it difficult for state of the art object detectors [Felzenszwalb et al., 2010; Schwartz et al., 2009] to reliably detect important objects in the scene: boat, humans and water – average precision scores reported in [Felzenszwalb et al., 2010] manages around 42% for humans and only 11% for boat over a dataset of almost 5000 images in 20 object categories. Yet, these images are *semantically* similar in terms of their high level description. Second, cognitive studies [Urgesi et al., 2006; Kourtzi, 2004] have proposed that inferring the action from static images (known as an "implied action") is often achieved by detecting the *pose* of humans in the image: the position of the limbs with respect to one another, under the assumption that a unique pose occurs for a unique action. Clearly, this assumption is weak as 1) similar actions may be represented by different poses due to the inherent dynamic nature of the action itself: e.g. walking a dog and 2) different actions may have the same pose: e.g. walking a dog versus running (Fig. 2(b)). The missing component here is whether the key object (dog) under interaction is considered. Recent works [Yao and Fei-Fei, 2010; Yang et al., 2010] that used poses for recognition of actions achieved 70% and 61% accuracy respectively under extremely limited testing conditions with only 5-6 action classes each. Finally, state of the art scene detectors [Oliva and Torralba, 2001; Torralba et al., 2003] need to have enough representative training examples of scenes from pre-defined scene classes for a classification to be successful – with a reported average precision of 83.7% tested over a dataset of 2600 images.

Addressing all these visual challenges is clearly a formidable task which is beyond the scope of this paper. Our focus instead is to show that with the addition of *language* to ground the noisy initial visual detections, we are able to improve the quality of the generated sentence as a faithful description of the image. In particular, we show that it is possible to avoid predicting actions directly from images – which is still unreliable – and to use the corpus instead to guide our predictions. Our proposed strategy is also *generic*, that is, we make no prior assumptions on the image domain considered. While other works (sec. 2) depend on strong annotations between images and text to ground their predictions (and to remove wrong sentences), we show that a large generic corpus is also able to provide the same grounding over larger domains of images. It represents a relatively new style of learning: distant supervision [Liang et al., 2009; Mann and Mccallum, 2007]. Here, we do not require "labeled" data containing images and captions but only separate data from each side. Another contribution is a computationally feasible way via dynamic programming to determine the most likely quadruplet $\mathcal{T}^* = \{n^*, v^*, s^*, p^*\}$ that describes the image for generating possible sentences.

2 Related Work

Recently, several works from the Computer Vision domain have attempted to use language to aid image scene understanding. [Kojima et al., 2000] used predefined production rules to describe actions in videos. [Berg et al., 2004] processed news captions to discover names associated with faces in the images, and [Jie et al., 2009] extended this work to associate poses detected from images with the verbs in the captions. Both approaches use annotated examples from a limited news caption corpus to learn a joint image-text model so that one can annotate new unknown images with textual information easily. Neither of these works have been tested on complex everyday images where the large variations of objects and poses makes it nearly impossible to learn a more general model. In addition, no attempt was made to generate a descriptive sentence from the learned model. The work of [Farhadi et al., 2010] attempts to "generate" sentences by first learning from a set of human annotated examples, and producing the *same* sentence if both images and sentence share common properties in terms of their triplets: (Nouns-Verbs-Scenes). No attempt was made to generate *novel* sentences from images beyond what has been annotated by humans. [Yao et al., 2010] has recently introduced a framework for parsing images/videos to textual description that requires significant annotated data, a requirement that our proposed approach avoids.

Natural language generation (NLG) is a long-standing problem. Classic approaches [Traum et al., 2003] are based on three steps: selection, planning and realization. A common challenge in generation problems is the question of: what is the input? Recently, approaches for generation have focused on formal specification inputs, such as the output of theorem provers [McKeown, 2009] or databases [Golland et al., 2010]. Most of the effort in those approaches has focused on selection and realization. We address a tangential problem that has not received much attention in the generation literature: how to deal with *noisy inputs*. In our case, the inputs themselves are often uncertain (due to misrecognitions by object/scene detectors) and the content selection and realization needs to take this uncertainty into account.

3 Our Approach

Our approach is summarized in Fig. 3. The input is a test image where we detect objects and scenes using trained detection algorithms [Felzenszwalb et al., 2010; Torralba et al., 2003]. To keep the framework computationally tractable, we limit the elements of the quadruplet (Nouns-Verbs-Scenes-Prepositions) to come from a finite set of objects $\mathcal{N}$, actions $\mathcal{V}$, scenes $\mathcal{S}$ and prepositions $\mathcal{P}$ classes that are commonly encountered. They are summarized in Table. 1. In addition, the sentence that is generated for each image is limited to at most two objects occurring in a unique scene.

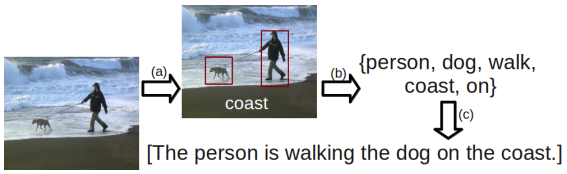


Figure 3: Overview of our approach. (a) Detect objects and scenes from input image. (b) Estimate optimal sentence structure quadruplet $\mathcal{T}^*$. (c) Generating a sentence from $\mathcal{T}^*$.

Denoting the current test image as I, the initial visual processing first detects objects $n \in \mathcal{N}$ and scenes $s \in \mathcal{S}$ using these detectors to compute $P_r(n|I)$ and $P_r(s|I)$, the probabilities that object n and scene s exist under I. From the observation that an action can often be predicted by its key objects, $N_k = \{n_1, n_2, \cdots, n_i\}, n_i \in \mathcal{N}$ that participate in the action, we use a trained Language model L_m to estimate $P_r(v|N_k)$. L_m is also used to compute $P_r(s|n, v)$, the predicted scene using the corpus given the object and verb; and $P_r(p|s)$, the predicted preposition given the scene. This process is repeated over all n, v, s, p where we used a modified HMM inference scheme to determine the most likely quadruplet: $\mathcal{T}^* = \{n^*, v^*, s^*, p^*\}$ that makes up the core sentence structure. Using the contents and structure of $\mathcal{T}^*$, an appropriate sentence is then generated that describes the image. In the following sections, we first introduce the image dataset used for testing followed by details of how these components are derived.

Objects $n \in \mathcal{N}$	Actions $v \in \mathcal{V}$	Scenes $s \in \mathcal{S}$	Preps $p \in \mathcal{P}$
'aeroplane' 'bicycle' 'bird'	'sit' 'stand' 'park'	'airport'	'in' 'at' 'above'
'boat' 'bottle' 'bus' 'car'	'ride' 'hold' 'wear'	'field'	'around' 'behind'
'cat' 'chair' 'cow' 'table'	'pose' 'fly' 'lie' 'lay'	'highway'	'below' 'beside'
'dog' 'horse', 'motorbike'	'smile' 'live' 'walk'	'lake' 'room'	'between'
'person' 'pottedplant'	'graze' 'drive' 'play'	'sky' 'street'	'before' 'to'
'sheep' 'sofa' 'train'	'eat' 'cover' 'train'	'track'	'under' 'on'
'tvmonitor'	'close' ...		

Table 1: The set of objects, actions (first 20), scenes and preposition classes considered

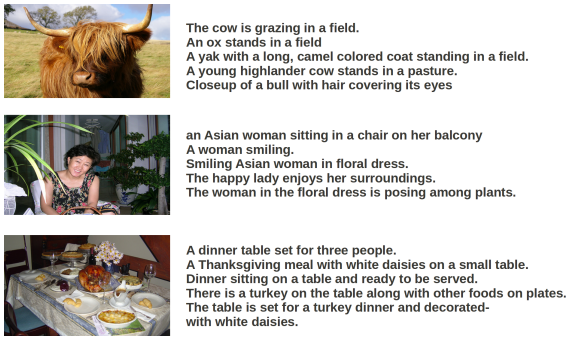
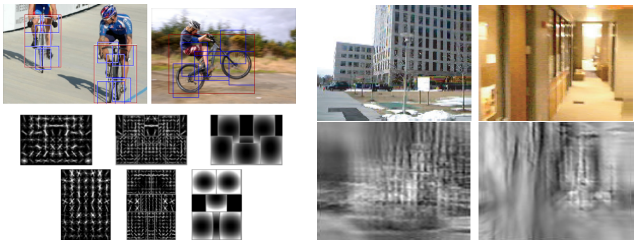


Figure 4: Samples of images with corresponding annotations from the UIUC scene description dataset.



(a) (b)

Figure 5: (a) [Top] The part based object detector from [Felzenszwalb et al., 2010]. [Bottom] The graphical model representation of an object, for e.g. a bike. (b) Examples of GIST gradients: (left) an outdoor scene vs (right) an indoor scene [Torralba et al., 2003].

3.1 Image Dataset

We use the *UIUC Pascal Sentence dataset*, first introduced in [Farhadi et al., 2010] and available online[1]. It contains 1000 images taken from a subset of the Pascal-VOC 2008 challenge image dataset and are hand annotated with sentences that describe the image by paid human annotators using Amazon Mechanical Turk. Fig. 4 shows some sample images with their annotations. There are 5 annotations per image, and each annotation is usually short – around 10 words long. We randomly selected 900 images (4500 sentences) as the learning corpus to construct the verb and scene sets, $\{\mathcal{V}, \mathcal{S}\}$ as described in sec. 3.3, and kept the remaining 100 images for testing and evaluation.

3.2 Object and Scene Detections from Images

We use the Pascal-VOC 2008 trained object detectors [Felzenszwalb et al., 2008] of 20 common everyday object classes that are defined in $\mathcal{N}$. Each of the detectors are essentially SVM classifiers trained on a large number of the objects' image representations from a large variety of sources. Although 20 classes may seem small, their existence in many natural images (e.g. humans, cars and plants) makes them particularly *important* for our task, since humans tend to describe these common objects as well. As object representations, the part-based descriptor of [Felzenszwalb et al., 2010] is used. This representation decomposes any object, e.g. a cow, into its constituent parts: head, torso, legs, which are shared by other objects in a hierarchical manner. At each level, image gradient orientations are computed. The relationship between each parts is modeled probabilistically using graphical models where parts are the nodes and the edges are the conditional probabilities that relate their spatial compatibility (Fig. 5(a)). For example, in a cow, the probability of finding the torso near the head is higher than finding the legs near the head. This model's intuition lies in the assumption that objects can be deformed but the relative position of each constituent parts should remain the same. We convert the object detection scores to probabilities using Platt's method [Lin et al., 2007] which is numerically more stable to obtain $P_r(n|I)$. The parameters of Platt's method are obtained by estimating the number of positives and negatives from the UIUC annotated dataset, from

[1] http://vision.cs.uiuc.edu/pascal-sentences/

which we determine the appropriate probabilistic threshold, which gives us approximately 50% recall and precision.

For detecting scenes defined in $\mathcal{S}$, we use the GIST-based scene descriptor of [Torralba et al., 2003]. GIST computes the windowed 2D Gabor filter responses of an input image. The responses of Gabor filters (4 scales and 6 orientations) encode the texture gradients that describe the *local* properties of the image. Averaging out these responses over larger spatial regions gives us a set of *global* image properties. These high dimensional responses are then reprojected to a low dimensional space via PCA, where the number of principal components are obtained empirically from training scenes. This representation forms the GIST descriptor of an image (Fig. 5(b)) which is used to train a set of SVM classifiers for each scene class in $\mathcal{S}$. Again, $P_r(s|I)$ is computed from the SVM scores using [Lin et al., 2007]. The set of common scenes defined in $\mathcal{S}$ is learned from the UIUC annotated data (sec. 3.3).

3.3 Corpus-Guided Predictions

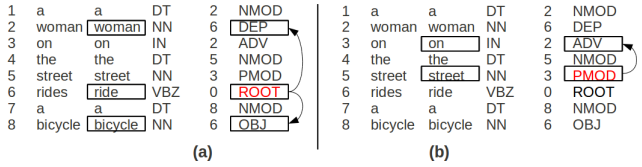


Figure 6: (a) Selecting the ROOT verb from the dependency parse `ride` reveals its subject `woman` and direct object `bicycle`. (b) Selecting the head noun (PMOD) as the scene `street` reveals ADV as the preposition `on`

Predicting Verbs: The key component of our approach is the trained language model L_m that predicts the most likely verb v, associated with the objects N_k detected in the image. Since it is possible that different verbs may be associated with varying number of object arguments, we limit ourselves to verbs that take on at most *two* objects (or more specifically two noun phrase arguments) as a simplifying assumption: $N_k = \{n_1, n_2\}$ where n_2 can be NULL. That is, n_1 and n_2 are the subject and direct objects associated with $v \in \mathcal{V}$. Using this assumption, we can construct the set of verbs, $\mathcal{V}$. To do this, we use human labeled descriptions of the training images from the UIUC Pascal-VOC dataset (sec. 3.1) as a learning corpus that allows us to determine the appropriate target verb set that is amenable to our problem. We first apply the CLEAR parser [Choi and Palmer, 2010] to obtain a dependency parse of these annotations, which also performs stemming of all the verbs and nouns in the sentence. Next, we process all the parses to select verbs which are marked as ROOT and check the existence of a subject (DEP) and direct object (PMOD, OBJ) that are linked to the ROOT verb (see Fig. 6(a)). Finally, after removing common "stop" verbs such as {`is, are, be`} we rank these verbs in terms of their occurrences and select the top 50 verbs which accounts for 87.5% of the sentences in the UIUC dataset to be in $\mathcal{V}$.

Object class $n \in \mathcal{N}$	Synonyms, $\langle n \rangle$
`bus`	`autobus charabanc double-decker jitney motorbus motorcoach omnibus passenger-vehicle schoolbus trolleybus streetcar ...`
`chair`	`highchair chaise daybed throne rocker armchair wheelchair seat ladder-back lawn-chair fauteuil ...`
`bicycle`	`bike wheel cycle velocipede tandem mountain-bike ...`

Table 2: Samples of synonyms for 3 object classes.

Next, we need to explain how n_1 and n_2 are selected from the 20 object classes defined previously in $\mathcal{N}$. Just as the 20 object classes are defined *visually* over several different kinds of specific objects, we expand n_1 and n_2 in their textual descriptions using *synonyms*. For example, the object class n_1=`aeroplane` should include the synonyms {`plane, jet, fighter jet, aircraft`}, denoted as $\langle n_1 \rangle$. To do this, we expand each object class using their corresponding WordNet synsets up to at most three hyponymns levels. Example synonyms for some of the classes are summarized in Table 2.

We can now compute from the Gigaword corpus [Graff, 2003] the probability that a verb exists given the detected nouns, $P_r(v|n_1, n_2)$. We do this by computing the log-likelihood ratio [Dunning, 1993] , λ_{nvn}, of *trigrams* $(\langle n_1 \rangle, v, \langle n_2 \rangle)$, computed from each sentence in the English Gigaword corpus [Graff, 2003]. This is done by extracting only the words in the corpus that are defined in $\mathcal{N}$ and $\mathcal{V}$ (in-

cluding their synonyms). This forms a *reduced* corpus sequence from which we obtain our target trigrams. For example, the sentence:

`the large brown dog chases a small young cat around the messy room, forcing the cat to run away towards its owner.`

will be reduced to the stemmed sequence `dog chase cat cat run owner`[2] from which we obtain the target trigram relationships: {`dog chase cat`}, {`cat run owner`} as these trigrams respect the (n_1, v, n_2) ordering. The log-likelihood ratios, λ_{nvn}, computed for all possible $(\langle n_1 \rangle, v, \langle n_2 \rangle)$ are then normalized to obtain $P_r(v|n_1, n_2)$. An example of ranked λ_{nvn} in Fig. 7(a) shows that λ_{nvn} predicts v that makes sense: with the most likely predictions near the top of the list.

Predicting Scenes: Just as an action is strongly related to the objects that participate in it, a scene can be predicted from the objects and verbs that occur in the image. For example, detecting N_k={`boat, person`} with v={`row`} would have predicted the scene s={`coast`}, since boats usually occur in water regions. To learn this relationship from the corpus, we use the UIUC dataset to discover what are the common scenes that should be included in $\mathcal{S}$. We applied the CLEAR dependency parse [Choi and Palmer, 2010] on the UIUC data and extracted all the head nouns (PMOD) in the PP phrases for this purpose and excluded those nouns with prepositions (marked as ADV) such as {`with, of`} which do not co-occur with scenes in general (see Fig. 6(b)). We then ranked the remaining scenes in terms of their frequency to select the top 8 scenes used in $\mathcal{S}$.

To improve recall and generalization, we expand each of the 8 scene classes using their WordNet synsets $\langle s \rangle$ (up to a max of three hyponymns levels). Similar to the procedure of predicting the verbs described above, we compute the log-likelihood ratio of ordered *bigrams*, $\{n, \langle s \rangle\}$ and $\{v, \langle s \rangle\}$: λ_{ns} and λ_{vs}, by reducing the corpus sentence to the target nouns, verbs and scenes defined in $\mathcal{N}, \mathcal{V}$ and $\mathcal{S}$. The probabilities $P_r(s|n)$ and $P_r(v|n)$ are then obtained by normalizing λ_{ns} and λ_{vs}. Under the assumption that the priors $P_r(n)$ and $P_r(v)$ are independent and applying Bayes rule, we can compute the probability that a scene co-occurs with the object and action, $P_r(s|n, v)$ by:

$$\begin{aligned} P_r(s|n,v) &= \frac{P_r(n,v|s)P_r(s)}{P_r(n,v)} \\ &= \frac{P_r(n|s)P_r(v|s)P_r(s)}{P_r(n)P_r(v)} \\ &\propto P_r(s|n) \times P_r(s|v) \end{aligned} \quad (1)$$

where the constant of proportionality is justified under the assumption that $P_r(s)$ is equiprobable for all s. (1) is computed for all nouns in N_k. As shown in Fig. 7(b), we are able to predict scenes that co-locate with reasonable correctness given the nouns and verbs.

Predicting Prepositions: It is straightforward to predict the appropriate prepositions associated with a given scene. When we construct $\mathcal{S}$ from the UIUC annotated data, we simply collect and rank all the associated prepositions (ADV) in the PP phrase of the dependency parses. We then select the top 12 prepositions used to define $\mathcal{P}$. Using $\mathcal{P}$, we then compute the log-likelihood ratio of ordered *bigrams*, $\{p, \langle s \rangle\}$ for prepositions that co-locate with the scene synonyms over the corpus. Normalizing λ_{ps} yields $P_r(p|s)$, the probability that a preposition co-locates with a scene. Examples of ranked λ_{ps} are shown in Fig. 7(c). Again, we see that reasonable predictions of p can be found.

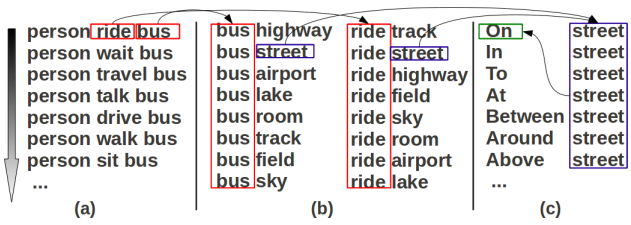


Figure 7: Example of how ranked log-likelihood values (in descending order) suggest a possible $\mathcal{T}$: (a) λ_{nvn} for $n_1 =$ `person`, $n_2 =$ `bus` predicts $v =$ `ride`. (b) λ_{ns} and λ_{vs} for $n =$ `bus`, $v =$ `ride` then jointly predicts $s =$ `street` and finally (c) λ_{ps} with $s =$ `street` predicts $p =$ `on`.

3.4 Determining $\mathcal{T}^*$ using HMM inference

Given the computed conditional probabilities: $P_r(n|I)$ and $P_r(s|I)$ which are observations from an input test image with the parameters of the trained language model, L_m:

[2] stemming is done using [Choi and Palmer, 2010]

$P_r(v|n_1, n_2), P_r(s|n, v), P_r(p|s)$, we seek to find the most likely sentence structure $\mathcal{T}^*$ by:

$$\begin{aligned}\mathcal{T}^* &= \arg\max_{n,v,s,p} P_r(\mathcal{T}|n, v, s, p) \\ &= \arg\max_{n,v,s,p}\{P_r(n_1|I)P_r(n_2|I)P_r(s|I)\times \\ &\qquad P_r(v|n_1, n_2)P_r(s|n, v)P_r(p|s)\} \quad (2)\end{aligned}$$

where the last equality holds by assuming independence between the visual detections and corpus predictions. Obviously a brute force approach to try all possible combinations to maximize eq. (2) will not be feasible due to the large number of possible combinations: $(20*21*8)*(50*20*20)*(8*20*50)*(12*8) \approx 5 \times 10^{13}$. A better solution is needed.

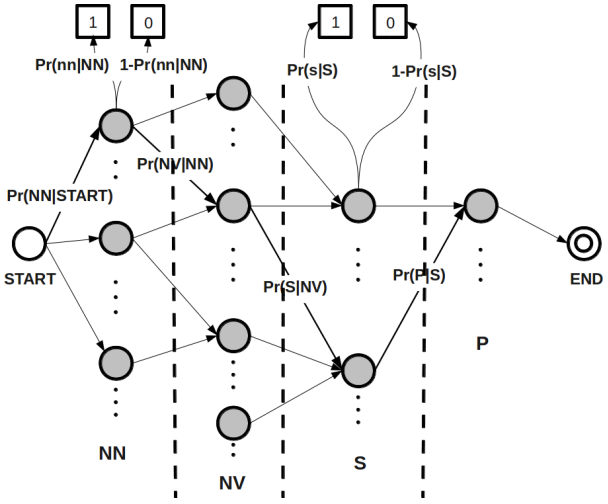


Figure 8: The HMM used for optimizing $\mathcal{T}$. The relevant transition and emission probabilities are also shown. See text for more details.

Our proposed strategy is to pose the optimization of $\mathcal{T}$ as a dynamic programming problem, akin to a Hidden Markov Model (HMM) where the hidden states are related to the (simplified) sentence structure we seek: $\mathcal{T} = \{n_1, n_2, s, v, p\}$, and the emissions are related to the observed detections: $\{n_1, n_2, s\}$ in the image if they exist. To simplify our notations, as we are concerned with object pairs we will write NN as the hidden states for all n_1, n_2 pairs and nn as the corresponding emissions (detections); and all object+verb pairs as hidden states NV. The hidden states are therefore denoted as: $\{\texttt{NN}, \texttt{NV}, \texttt{S}, \texttt{P}\}$ with values taken from their respective word classes from Table 1. The emission states are $\{\texttt{nn}, \texttt{s}\}$ with binary values: 1 if the detections occur or 0 otherwise. The full HMM is summarized in Fig. 8. The rationale for using a HMM is that we can reuse all previous computation of the probabilities at each level to compute the required probabilities at the current level. From START, we assume all object pair detections are equiprobable: $P_r(\texttt{NN}|\texttt{START}) = \frac{1}{|\mathcal{N}|*(|\mathcal{N}|+1)}$ where we have added an additional NULL value for objects (at most 1). At each NN, the HMM emits a detection from the image and by independence we have: $P_r(\texttt{nn}|\texttt{NN}) = P_r(n_1|I)P_r(n_2|I)$. After NN, the HMM transits to the corresponding verb at state NV with $P_r(\texttt{NV}|\texttt{NN}) = P_r(v|n_1, n_2)$ obtained from the corpus statistic[3]. As no action detections are performed on the image, NV has no emissions. The HMM then transits from NV to S with $P_r(\texttt{S}|\texttt{NV}) = P_r(s|n, v)$ computed from the corpus which emits the scene detection score from the image: $P_r(\texttt{s}|\texttt{S}) = P_r(s|I)$. From S, the HMM transits to P with $P_r(\texttt{P}|\texttt{S}) = P_r(p|s)$ before reaching the END state.

Comparing the HMM with eq. (2), one can see that all the corpus and detection probabilities are accounted for in the transition and emission probabilities respectively. Optimizing $\mathcal{T}$ is then equivalent to finding the best (most likely) path through the HMM given the image observations using the Viterbi algorithm which can be done in $O(10^5)$ time which is significantly faster than the naive approach. We show in Fig. 9 (right-upper) examples of the top viterbi paths that produce $\mathcal{T}^*$ for four test images.

Note that the proposed HMM is suitable for generating sentences that contain the core components defined in $\mathcal{T}$ which produces a sentence of the form NP−VP−PP, which we will show in sec. 4 is sufficient for the task of generating sentences for describing images. For more complex sentences with more components: such as adjectives or adverbs, the HMM can be easily extended with similar computations derived from the corpus.

3.5 Sentence Generation

Given the selected sentence structure $\mathcal{T} = \{n_1, n_2, v, s, p\}$, we generate sentences using the

[3]each verb, v, in NV will have 2 entries with the same value, one for each noun.

{aeroplane,fly,airport,at}
the aeroplane is flying at the airport.

{person,motorbike,ride,field,in}
the person is riding the motorbike in the field.

{person,bicycle,ride,street,on}
the person is riding the bicycle on the street.

{person,table,sit,room,in}
three people are sitting at the table in the room.

Figure 9: Four test images (left) and results. (Right-upper): Sentence structure $\mathcal{T}^*$ predicted using Viterbi and (Right-lower): Generated sentences. Words marked in red are considered to be incorrect predictions. Complete results are available at `http://www.umiacs.umd.edu/~yzyang/sentence_generateOut.html`.

following strategy for each component:

1) We add in appropriate determiners and cardinals: `the, an, a, CARD`, based on the content of n_1,n_2 and s. For e.g., if $n_1 = n_2$, we will use `CARD=two`, and modify the nouns to be in the plural form. When several possible choices are available, a random choice is made that depends on the object detection scores: `the` is preferred when we are confident of the detections while `an, a` is preferred otherwise.

2) We predict the most likely preposition inserted between the verbs and nouns learned from the Gigaword corpus via $P_r(p|v,n)$ during sentence generation. For example, our method will pick the preposition `at` between verb `sit` and noun `table`.

3) The verb v is converted to a form that agrees with in number with the nouns detected. The present gerund form is preferred such as `eating, drinking, walking` as it conveys that an action is being performed in the image.

4) The sentence structure is therefore of the form: `NP-VP-PP` with variations when only one object or multiple detections of the same objects are detected. A special case is when *no* objects are detected (below the predefined threshold). No verbs can be predicted as well. In this case, we simply generate a sentence that describes the *scene* only: for e.g. `This is a coast, This is a field`. Such sentences account for 20% of the entire UIUC testing dataset which are scored lower in our evaluation metrics (sec. 4.1) since they do not fully *describe* the image content in terms of the objects and actions.

Some examples of sentences generated using this strategy are shown in Fig. 9(right-lower).

4 Experiments

We performed several experiments to evaluate our proposed approach. The different metrics used for evaluation and comparison are also presented, followed by a discussion of the experimental results.

4.1 Sentence Generation Results

Three experiments are performed to evaluate the effectiveness of our approach. As a baseline, we simply generated $\mathcal{T}^*$ *directly* from images without using the corpus. There are two variants of this baseline where we seek to determine if listing *all* objects in the image is crucial for scene description. $\mathcal{T}_{b1}$ is a baseline that uses *all* possible objects and scene detected: $\mathcal{T}_{b1} = \{n_1, n_2, \cdots, n_m, s\}$ and our sentence will be of the form: {`Object 1, object 2 and object 3 are IN the scene.`} and we simply selected `IN` as the only admissible preposition. For the second baseline, $\mathcal{T}_{b2}$, we limit the number of objects to just any two: $\mathcal{T}_{b2} = \{n_1, n_2, s\}$ and the sentence generated will be of the form {`Object 1 and object 2 are IN the scene`}. In the second experiment, we applied the HMM strategy described above but made all transition probabilities *equiprobable*, removing the effects of the corpus, and producing a sentence structure which we denote as $\mathcal{T}^*_{eq}$. The third experiment produces the full $\mathcal{T}^*$ with transition probabilities learned from the corpus. All experiments were performed on the 100 unseen testing images from the UIUC dataset and we used only the most likely (top) sentence generated for all evaluation.

We use two evaluation metrics as a measure of the accuracy of the generated sentences: 1) ROUGE-1 [Lin and Hovy, 2003] precision scores and 2) *Relevance* and *Readability* of the generated sentences. ROUGE-1 is a recall based metric that is commonly used to measure the effectiveness of text summarization. In this work, the short descriptive sentence of an image can be viewed as summarizing the image

content and ROUGE-1 is able to capture how well this sentence can describe the image by comparing it with the human annotated ground truth of the UIUC dataset. Due to the short sentences generated, we did not consider other ROUGE metrics (ROUGE-2, ROUGE-SU4) which captures fluency and is not an issue here.

Experiment	R_1,(length)	Relevance	Readability
Baseline 1, $\mathcal{T}_{b1}^*$	0.35,(8.2)	$\mathbf{2.84} \pm 1.40$	3.64 ± 1.20
Baseline 2, $\mathcal{T}_{b2}^*$	0.39,(6.8)	2.14 ± 1.13	3.94 ± 0.91
HMM no corpus, $\mathcal{T}_{eq}^*$	0.42,(6.5)	2.44 ± 1.25	3.88 ± 1.18
Full HMM, $\mathcal{T}^*$	**0.44**,(6.9)	2.51 ± 1.30	$\mathbf{4.10} \pm 1.03$
Human Annotation	0.68,(10.1)	4.91 ± 0.29	4.77 ± 0.42

Table 3: Sentence generation evaluation results with human gold standard. Human R_1 scores are averaged over the 5 sentences using a leave one out procedure. Values in bold are the top scores.

A main shortcoming of using ROUGE-1 is that the generated sentences are compared only to a finite set of human labeled ground truth which obviously does not capture all possible sentences that one can generate. In other words, ROUGE-1 does not take into account the fact that sentence generation is innately a *creative* process, and a better recall metric will be to ask humans to judge these sentences. The second evaluation metric: Relevance and Readability is therefore proposed as an empirical measure of how much the sentence: 1) conveys the image content (relevance) in terms of the objects, actions and scene predicted and 2) is grammatically correct (readability). We engaged the services of Amazon Mechanical Turks (AMT) to judge the generated sentences based on a discrete scale ranging from 1–5 (low relevance/readability to high relevance/readability). The averaged results of ROUGE-1, R_1 and mean length of the sentences with the Relevance+Readability scores for all experiments are summarized in Table 3. For comparison, we also asked the AMTs to judge the ground truth sentences as well.

4.2 Discussion

The results reported in Table 3 reveals both the strengths and some shortcomings of the approach which we will briefly discuss here. Firstly, the R_1 scores indicate that based on a purely summarization (unigram-overlap) point of view, the proposed approach of using the HMM to predict $\mathcal{T}^*$ achieves the best results compared to all other approaches with $R_1 = 0.44$. This means that our sentences are the closest in agreement with the human annotated ground truth, correctly predicting the sentence structure components. In addition sentences generated by $\mathcal{T}^*$ are also succinct: with an average length of 6.9 words per sentence. However, we are still some way off the human gold standard since we do not predict other parts-of-speech such as adjectives and adverbs. Given this fact, our proposed approach performance is comparable to other state of the art summarization work in the literature [Bonnie and Dorr, 2004].

Next, we consider the Relevance+Readability metrics based on human judges. Interestingly, the first baseline, $\mathcal{T}_{b1}^*$ is considered the most relevant description of the image and the least readable at the same time. This is most likely due to the fact that this recall oriented strategy will almost certainly describe some objects but the lack of any verb description; and longer sentences that average 8.2 words per sentence, makes it less readable. It is also possible that humans tend to penalize less irrelevant objects compared to missing objects, and further evaluations are necessary to confirm this. Since $\mathcal{T}_{b2}^*$ is limited to two objects just like the proposed HMM, it is a more suitable baseline for comparison. Clearly, the results show that adding the HMM to predict the optimal sentence structure increases the relevance of the produced sentence. Finally, in terms of *readability*, $\mathcal{T}^*$ generates the most readable sentences, and this is achieved by leveraging on the corpus to guide our predictions of the most *reasonable* nouns, verbs, scenes and prepositions that agree with the detections in the image.

5 Future Work

In this work, we have introduced a computationally feasible framework that integrates visual perception together with semantic grounding obtained from a large textual corpus for the purpose of generating a descriptive sentence of an image. Experimental results show that our approach produces sentences that are both relevant and readable. There are, however, instances where our strategy fails to predict the ap-

propriate verbs or nouns (see Fig. 9). This is due to the fact that object/scene detections can be wrong and noise from the corpus itself remains a problem. Compared to human gold standards, therefore, much work still remains in terms of detecting these objects and scenes with high precision. Currently, at most two object classes are used to generate simple sentences which was shown in the results to have penalized the relevance score of our approach. This can be addressed by designing more complex HMMs to handle larger numbers of object and verb classes. Another interesting direction of future work would be to detect *salient* objects, learned from training image+corpus or eye-movement data, and to verify if these objects aid in improving the descriptive sentences we generate. Another potential application

Figure 10: Images retrieved from 3 verbal search terms: `ride,sit,fly`.

of representing images using $\mathcal{T}^*$ is that we can easily sort and retrieve images that are similar in terms of their *semantic* content. This would enable us to retrieve, for example, more relevant images given a verbal search query such as {`ride,sit,fly`}, returning images where these verbs are found in $\mathcal{T}^*$. Some results of retrieved images based on their verbal components are shown in Fig. 10: many images with dissimilar visual content are correctly classified based on their semantic meaning.

6 Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1035542. In addition, the support of the European Union under the Cognitive Systems program (project POETICON) and the National Science Foundation under the Cyberphysical Systems Program, is gratefully acknowledged.

References

Berg, T. L., Berg, A. C., Edwards, J., and Forsyth, D. A. (2004). Who's in the picture? In *NIPS*.

Bonnie, D. Z. and Dorr, B. (2004). Bbn/umd at duc-2004: Topiary. In *In Proceedings of the 2004 Document Understanding Conference (DUC 2004) at NLT/NAACL 2004*, pages 112–119.

Choi, J. D. and Palmer, M. (2010). Robust constituent-to-dependency conversion for english. In *Proceedings of the 9th International Workshop on Treebanks and Linguistic Theories*, pages 55–66, Tartu, Estonia.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2008). The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results.

Farhadi, A., Hejrati, S. M. M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., and Forsyth, D. A. (2010). Every picture tells a story: Generating sentences from images. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *ECCV (4)*, volume 6314 of *Lecture Notes in Computer Science*, pages 15–29. Springer.

Felzenszwalb, P. F., Girshick, R. B., and McAllester, D. (2008). Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D. A., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645.

Golland, D., Liang, P., and Klein, D. (2010). A game-theoretic approach to generating spatial descriptions. In *Proceedings of EMNLP*.

Graff, D. (2003). English gigaword. In *Linguistic Data Consortium, Philadelphia, PA*.

Jie, L., Caputo, B., and Ferrari, V. (2009). Who's doing what: Joint modeling of names and verbs for simultaneous face and pose annotation. In NIPS, editor, *Advances in Neural Information Processing Systems*, NIPS. NIPS.

Kojima, A., Izumi, M., Tamura, T., and Fukunaga, K. (2000). Generating natural language description of human behavior from video images. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 728 –731 vol.4.

Kourtzi, Z. (2004). But still, it moves. *Trends in Cognitive Sciences*, 8(2):47 – 49.

Liang, P., Jordan, M. I., and Klein, D. (2009). Learning from measurements in exponential families. In *International Conference on Machine Learning (ICML)*.

Lin, C. and Hovy, E. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACLHLT*.

Lin, H.-T., Lin, C.-J., and Weng, R. C. (2007). A note on platt's probabilistic outputs for support vector machines. *Mach. Learn.*, 68:267–276.

Mann, G. S. and Mccallum, A. (2007). Simple, robust, scalable semi-supervised learning via expectation regularization. In *The 24th International Conference on Machine Learning*.

McKeown, K. (2009). Query-focused summarization using text-to-text generation: When information comes from multilingual sources. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation (UCNLG+Sum 2009)*, page 3, Suntec, Singapore. Association for Computational Linguistics.

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175.

Schwartz, W., Kembhavi, A., Harwood, D., and Davis, L. (2009). Human detection using partial least squares analysis. In *International Conference on Computer Vision*.

Torralba, A., Murphy, K. P., Freeman, W. T., and Rubin, M. A. (2003). Context-based vision system for place and object recognition. In *ICCV*, pages 273–280. IEEE Computer Society.

Traum, D., Fleischman, M., and Hovy, E. (2003). Nl generation for virtual humans in a complex social environment. In *In Proceedings of he AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, pages 151–158.

Urgesi, C., Moro, V., Candidi, M., and Aglioti, S. M. (2006). Mapping implied body actions in the human motor system. *J Neurosci*, 26(30):7942–9.

Yang, W., Wang, Y., and Mori, G. (2010). Recognizing human actions from still images with latent poses. In *CVPR*.

Yao, B. and Fei-Fei, L. (2010). Grouplet: a structured image representation for recognizing human and object interactions. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA.

Yao, B., Yang, X., Lin, L., Lee, M. W., and Zhu, S.-C. (2010). I2t: Image parsing to text description. *Proceedings of the IEEE*, 98(8):1485 –1508.

Corroborating Text Evaluation Results with Heterogeneous Measures

Enrique Amigó † Julio Gonzalo † Jesús Giménez ‡ Felisa Verdejo†

† UNED, Madrid
{enrique,julio,felisa}@lsi.uned.es

‡ UPC, Barcelona
{jgimenez}@lsi.upc.edu

Abstract

Automatically produced texts (e.g. translations or summaries) are usually evaluated with n-gram based measures such as BLEU or ROUGE, while the wide set of more sophisticated measures that have been proposed in the last years remains largely ignored for practical purposes. In this paper we first present an in-depth analysis of the state of the art in order to clarify this issue. After this, we formalize and verify empirically a set of properties that every text evaluation measure based on similarity to human-produced references satisfies. These properties imply that corroborating system improvements with additional measures always increases the overall reliability of the evaluation process. In addition, the greater the *heterogeneity* of the measures (which is measurable) the higher their combined reliability. These results support the use of heterogeneous measures in order to consolidate text evaluation results.

1 Introduction

The automatic evaluation of textual outputs is a core issue in many Natural Language Processing (NLP) tasks such as Natural Language Generation, Machine Translation (MT) and Automatic Summarization (AS). State-of-the-art automatic evaluation methods all operate by rewarding similarities between automatically-produced candidate outputs and manually-produced reference solutions, so-called human references or models.

Over the last decade, a wide variety of measures, based on different quality assumptions, have been proposed. Recent work suggests exploiting external knowledge sources and/or deep linguistic annotation, and measure combination (see Section 2). However, original measures based on lexical matching, such as BLEU (Papineni et al., 2001a) and ROUGE (Lin, 2004) are still preferred as de facto standards in MT and AS, respectively. There are, in our opinion, two main reasons behind this fact. First, the use of a common measure certainly allows researchers to carry out objective comparisons between their work and other published results. Second, the advantages of novel measures are not easy to demonstrate in terms of correlation with human judgements.

Our goal is not to answer which is the most reliable metric or to propose yet another novel measure. Rather than this, we first analyze in depth the state of the art, concluding that it is not easy to determine the reliability of a measure. In absence of a clear proof of the advantages of novel measures, system developers naturally tend to prefer well-known standard measures. Second, we formalize and check empirically two intrinsic properties that any evaluation measure based on similarity to human-produced references satisfies. Assuming that a measure satisfies a set of basic formal constraints, these properties imply that corroborating a system comparison with additional measures always increases the overall reliability of the evaluation process, even when the added measures have a low correlation with human judgements. In most papers, evaluation results are corroborated with similar n-gram based measures (eg. BLEU and ROUGE). However, according to our second property, the greater the *heterogeneity* of

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 455–466,
Edinburgh, Scotland, UK, July 27–31, 2011.

the measures (which is measurable) the higher their reliability. The practical implication is that, corroborating evaluation results with measures based on higher linguistic levels increases the heterogeneity, and therefore, the reliability of evaluation results.

2 State of the Art

2.1 Individual measures

Among NLP disciplines, MT probably has the widest set of automatic evaluation measures. The dominant approach to automatic MT evaluation is, today, based on lexical metrics (also called n-gram based metrics). These metrics work by rewarding lexical similarity between candidate translations and a set of manually-produced reference translations. Lexical metrics can be classified according to how they compute similarity. Some are based on edit distance, e.g., WER (Nießen et al., 2000), PER (Tillmann et al., 1997), and TER (Snover et al., 2006). Other metrics are based on computing lexical precision, e.g., BLEU (Papineni et al., 2001b) and NIST (Doddington, 2002), lexical recall, e.g., ROUGE (Lin and Och, 2004a) and CDER (Leusch et al., 2006), or a balance between the two, e.g., GTM (Melamed et al., 2003; Turian et al., 2003b), METEOR (Banerjee and Lavie, 2005), BLANC (Lita et al., 2005), SIA (Liu and Gildea, 2006), MAXSIM (Chan and Ng, 2008), and O_l (Giménez, 2008).

The lexical measure BLEU has been criticized in many ways. Some drawbacks of BLEU are the lack of interpretability (Turian et al., 2003a), the fact that it is not necessary to increase BLEU to improve systems (Callison-burch and Osborne, 2006), the overscoring of statistical MT systems (Le and Przybocki, 2005), the low reliability over rich morphology languages (Homola et al., 2009), or even the fact that a poor system translation of a book can obtain higher BLEU results than a manually produced translation (Culy and Riehemann, 2003).

The reaction to these criticisms has been focused on the development of more sophisticated measures in which candidate and reference translations are automatically annotated and compared at different linguistic levels. Some of the features employed include parts of speech (Popovic and Ney, 2007; Giménez and Màrquez, 2007), syntactic dependencies (Liu and Gildea, 2005; Giménez and Màrquez, 2007; Owczarzak et al., 2007a; Owczarzak et al., 2007b; Owczarzak et al., 2008; Chan and Ng, 2008; Kahn et al., 2009), CCG parsing (Mehay and Brew, 2007), syntactic constituents (Liu and Gildea, 2005; Giménez and Màrquez, 2007), named entities (Reeder et al., 2001; Giménez and Màrquez, 2007), semantic roles (Giménez and Màrquez, 2007), discourse representations (Giménez, 2008), and textual entailment features (Padó et al., 2009). In general, when a higher linguistic level is incorporated, linguistic features at lower levels are preserved.

The proposals for summarization evaluation are less numerous. Some proposals for AS tasks are based on syntactic units (Tratz and Hovy, 2008), dependency triples (Owczarzak, 2009) or convolution kernels (Hirao et al., 2005) which reported some reliability improvement over ROUGE in terms of correlation with human judgements.

In general, however, it is not easy to determine clearly the contribution of deeper linguistic knowledge in those proposals. In the case of MT, improvements versus BLEU have been reported (Liu and Gildea, 2005; Kahn et al., 2009), but not over a more elaborated metric such as METEOR (Mehay and Brew, 2007; Chan and Ng, 2008). Besides, controversial results on their performance at sentence vs system level have been reported in shared evaluation tasks (Callison-Burch et al., 2008; Callison-Burch et al., 2009; Callison-Burch et al., 2010).

2.2 Combined measures

Several researchers have suggested integrating heterogeneous measures. Some of them optimize the measure combination function according to the metric's ability to emulate the behavior of human assessors (i.e., correlation with human assessments). For instance, using linear combinations (Padó et al., 2009; Liu and Gildea, 2007; Giménez and Màrquez, 2008), Decision Trees (Akiba et al., 2001; Quirk, 2004), regression based algorithms (Paul et al., 2007; Albrecht and Hwa, 2007a; Albrecht and Hwa, 2007b) or a variety of supervised machine learning algorithms(Quirk et al., 2005; Corston-Oliver et al., 2001; Kulesza and Shieber, 2004; Gamon et al., 2005; Amigó et al., 2005).

Some of these works report evidence on the contribution of combining heterogeneous measures. For instance, Albrecht and Hwa included syntax-based

measures together with lexical measures, outperforming other combination schemes (Albrecht and Hwa, 2007a; Albrecht and Hwa, 2007b). Liu and Gildea, after examining the contribution of each component metric, found that *"metrics showing different properties of a sentence are more likely to make a good combined metric"*(Liu and Gildea, 2007). Akiba et al., which combined multiple edit-distance features based on lexical, morphosyntactic and lexical semantic information, observed that their approach improved single editing distance for several data sets (Akiba et al., 2001). More evidence was provided by Corston and Oliver. They showed that results on the task of discriminating between manual and automatic translations improve when combining linguistic and n-gram based features. In addition, they showed that this mixed combination improved over the combination of linguistic or n-gram based measures alone (Corston-Oliver et al., 2001). (Padó et al., 2009) reported a reliability improvement by including measures based on textual entailment in the set. In (Giménez and Màrquez, 2008), a simple arithmetic mean of scores for combining measures at different linguistic levels was applied with remarkable results in recent shared evaluation tasks (Callison-Burch et al., 2010).

2.3 Meta-evaluation criteria

Meta-evaluation methods have been gradually introduced together with evaluation measures. For instance, Papineni et al. (2001b) evaluated the reliability of the BLEU metric according to its ability to emulate human assessors, as measured in terms of Pearson correlation with human assessments of adequacy and fluency at the document level. The measure NIST (Doddington, 2002) was meta-evaluated also in terms of correlation with human assessments, but over different document sources and for a varying number of references and segment sizes. Melamed et al. (2003) argued, at the time of introducing the GTM metric, that Pearson correlation coefficients can be affected by scale properties. They suggested using the non-parametric Spearman correlation coefficients instead. Lin and Och meta-evaluated ROUGE over both Pearson and Spearman correlation over a wide set of metrics, including NIST, WER, PER, and variants of ROUGE, BLEU and GTM. They obtained similar results in both cases (Lin and Och, 2004a). Banerjee and Lavie (2005) argued that the reliability of metrics at the document level can be due to averaging effects but might not be robust across sentence translations. In order to address this issue, they computed the translation-by-translation correlation with human assessments (i.e., correlation at the sentence level).

However, correlation with human judgements is not enough to determine the reliability of measures. First, correlation at sentence level (unlike correlation at system level) tends to be low and difficult to interpret. Second, correlation at system and segment levels can produce contradictory results. In (Amigó et al., 2009) it is observed that higher linguistic levels in measures increases the correlation with human judgements at the system level at the cost of correlation at the segment level. As far as we know, a clear explanation for these phenomena has not been provided yet.

Third, a high correlation at system level does not ensure a high reliability. Culy and Rieheman observed that, although BLEU can achieve a high correlation at system level in some test suites, it over-scores a poor automatic translation of "Tom Sawyer" against a human produced translation (Culy and Riehemann, 2003). This meta-evaluation criterion based on the ability to discern between manual and automatic translations have been referred to as *human likeness* (Amigó et al., 2006), in contrast to correlation with human judgements which is referred to as *human acceptability*. Examples of meta-measures based on this criterion are ORANGE (Lin and Och, 2004b) and KING (Amigó et al., 2005). In addition, many of the approaches to metric combination described in Section 2.2 take human likeness as the optimization criterion (Corston-Oliver et al., 2001; Kulesza and Shieber, 2004; Gamon et al., 2005). The main advantage of meta-evaluation based on human likeness is that, since human assessments are not required, metrics can be evaluated over larger test beds. However, the meta-evaluation in terms of *human likeness* is difficult to interpret.

2.4 The use of evaluation measures

In general, the state of the art includes a wide set of results that show the drawbacks of n-gram based measures as BLEU, and a wide set of proposals for new single and combined measures which are meta-

evaluated in terms of human acceptability (i.e., their ability to emulate human judges, typically measured in terms of correlation with human judgements) or human-likeness (i.e., their ability to discern between automatic and human translations) (Amigó et al., 2006). However, the original measures BLEU and ROUGE are still preferred.

We believe that one of the reasons is the lack of an in-depth study on to what extent providing additional evaluation results with other metrics contributes to the reliability of such results. The state of the art suggests that the use of heterogeneous measures can improve the evaluation reliability. However, as far as we know, there is no comprehensive analysis on the contribution of novel measures when corroborating evaluation results with additional measures.

3 Similarity Based Evaluation Measures

In general, automatic evaluation measures applied in tasks like MT or AS are similarity measures between system outputs and human references. These measures are related with precision, recall or overlap over specific types of linguistic units. For instance, ROUGE measures n-gram recall. Other measures that work at higher linguistic levels apply precision, recall or overlap of linguistic components such as dependency relations, grammatical categories, semantic roles, etc.

In order to delimit our hypothesis, let us first define what is a similarity measure in this context. Unfortunately, as far as we know, there is no formal concept covering the properties of current evaluation similarity measures. A close concept is that of *"metric"* or *"distance function"*. But, actually, measures such as ROUGE or BLEU are not proper *"metrics"*, because they do not satisfy the *symmetry* and the *triangle inequality* properties. Therefore, we need a new definition.

Being Ω the universe of system outputs s and gold-standards g, we assume that a *similarity measure*, in our context, is a function $x : \Omega^2 \longrightarrow \Re$ such that there exists a decomposition function $f : \Omega \longrightarrow \{e_1..e_n\}$ (e.g., words or other linguistic units or relationships) satisfying the following constraints: (i) maximum similarity is achieved only when then the decomposition of the system output resembles exactly the gold-standard decomposition; and (ii) growing overlap or removing non overlapped elements implies growing x. Formally, if x ranges from 0 to 1:

$$f(s) = f(g) \leftrightarrow x(s,g) = 1$$
$$(f(s) = f(s') \cup \{e \in f(g) \setminus f(s')\}) \rightarrow x(s,g) > x(s',g)$$
$$(f(s) = f(s') - \{e \in f(s') \setminus f(g)\}) \rightarrow x(s,g) > x(s',g)$$

For instance, a random function and the reversal of a similarity funtion ($f'(s) = \frac{1}{f(s)}$) do not satisfy these constraints. While the F measure over *Precision* and *Recall* satisfies these constraints[1], precision and recall in isolation do not satisfy all of them: maximum recall can be achieved without resembling the goldstandard text decomposition; and maximum precision can be achieved with only a few overlapped elements.

BLEU (Papineni et al., 2001a) computes the n-gram precision while the metric ROUGE (Lin and Och, 2004a) computes the n-gram recall. However, in general, both metrics satisfy all the constraints, given that BLEU includes a brevity penalty and ROUGE penalizes or limits the system output length. The measure METEOR creates an alignment between the two strings (Banerjee and Lavie, 2005). This overlap-based measure satisfies also the previous constraints. Measures based on edit distance over n-grams (Tillmann et al., 1997; Nießen et al., 2000) or other linguistic units (Akiba et al., 2001; Popovic and Ney, 2007) match also our definition of similarity measure. The editing distance is minimum when the two compared text are equal. The more the evaluated text contains elements from the gold-standard the more the editing distance is reduced (higher similarity). The word ordering can be also expressed in terms of a decomposition function. A similar reasoning applies to every relevant measure in the state-of-the art.

4 Data Sets and Measures

4.1 Data sets

In this paper, we provide empirical results for MT and AS. For MT, we use the data sets from the Arabic-to-English (AE) and Chinese-to-English (CE) NIST MT Evaluation campaigns in 2004 and

[1]There is an exception. In an extreme case, when recall is zero, removing non overlapped elements does not modify the F measure.

	$\mathbf{AE_{2004}}$	$\mathbf{CE_{2004}}$	$\mathbf{AE_{2005}}$	$\mathbf{CE_{2005}}$
#human-references	5	5	5	4
#systems	5	10	7	10
#system-outputs-assessed	5	10	6	5
#system-outputs	1,353	1,788	1,056	1,082
#outputs-assessed per-system	347	447	266	272

Table 1: Description of the test beds from 2004 and 2005 NIST MT evaluation campaigns used in the experiments throughout the paper.

	DUC 2005	DUC 2006
#human-references	3-4	3-4
#systems	32	35
#system-outputs-assessed	32	35
#system-outputs	50	50
#outputs-assessed per-system	50	50

Table 2: Description of the test beds from 2005 and 2006 DUC evaluation campaigns used in the experiments throughout the paper.

2005[2]. Both include two translations exercises: for the 2005 campaign we contacted each participant individually and asked for permission to use their data[3]. In our experiments, we take the sum of adequacy and fluency, both in a 1-5 scale, as a global measure of quality (LDC, 2005). Thus, human assessments are in a 2-10 scale. For AS, we have used the AS test suites developed in the DUC 2005 and DUC 2006 evaluation campaigns[4]. This AS task was to generate a question focused summary of 250 words from a set of 25-50 documents to a complex question. Summaries were evaluated according to several criteria. Here, we will consider the responsiveness judgements, in which the quality score was an integer between 1 and 5. See Tables 1 and 2 for a brief quantitative description of these test beds.

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

[3] We are grateful to a number of groups and companies who responded positively: University of Southern California Information Sciences Institute (ISI), University of Maryland (UMD), Johns Hopkins University & University of Cambridge (JHU-CU), IBM, University of Edinburgh, University of Aachen (RWTH), National Research Council of Canada (NRC), Chinese Academy of Sciences Institute of Computing Technology (ICT), Instituto Trentino di Cultura - Centro per la Ricerca Scientifica e Tecnologica(ITC-IRST), MITRE.

[4] http://duc.nist.gov/

4.2 Measures

As for evaluation measures, for MT we have used a rich set of 64 measures provided within the ASIYA Toolkit (Giménez and Màrquez, 2010)[5]. This includes measures operating at different linguistic levels: lexical, syntactic, and semantic. At the lexical level this set includes variants of 8 measures employed in the state of the art: BLEU, NIST, GTM, METEOR, ROUGE, WER, PER and TER. In addition, we have included a basic measure O_l that computes the lexical overlap without considering word ordering. All these measures have similar granularity. They use n-grams of a varying length as the basic unit with additional information provided by linguistic tools. The underlying similarity criteria include precision, recall, overlap, or edit rate, and the decomposition functions include words, dependency tree nodes (DP_HWC, DP-Or, etc.), constituency parsing (CP-STM), discourse roles (DR-Or), semantic roles (SR-Or), named entities, etc. Further details on the measure set may be found in the ASIYA technical manual (Giménez and Màrquez, 2010).

According to our computations, our measures cover high and low correlations at both levels. Correlation at system level spans between 0.63 and 0.95. Correlations at sentence level ranges from 0.18 up to 0.54. We will discriminate between two subsets of

[5] http://www.lsi.upc.edu/~nlp/Asiya

measures. The first one includes those that decompose the text into words, n-grams, stems or lexical semantic tags. This set includes BLEU, ROUGE, NIST, GTM, PER and WER families. We will refer to them as "lexical" measures. The second set are those that consider deeper linguistic levels such as parts of speech, syntactic dependencies, syntactic constituents, etc. We will refer to them as "linguistic" measures.

In the case of automatic summarization (AS), we have employed the standard variants of ROUGE (Lin, 2004). These 7 measures are ROUGE-$\{1..4\}$, ROUGE-SU, ROUGE-L and ROUGE-W. In addition we have included the reversed precision version for each variant and the F measure of both. Notice that the original ROUGE measures are oriented to recall. In total, we have 21 measures for the summarization task. All of them are based on n-gram overlap.

5 Additive reliability

As discussed in Section 2, a number of recent publications address the problem of measure combination with successful results, specially when heterogeneous measures are combined. The following property clarifies this issue and justifies the use of heterogeneous measures when corroborating evaluation results. It asserts that *the reliability of system improvements always increases when the evaluation result is corroborated by an additional similarity measure, regardless of the correlation achieved by the additional measure in isolation*.

For the sake of clarity, in the rest of the paper, we will denote the similarity $x(s, g)$ between system output s and human reference g by $x(s)$. The quality of a system output s will be referred to as $Q(s)$. Let us define the *reliability* $R(X)$ of a measure set as the probability of a real improvement (as measured by human judges) when a score improvement is observed simultaneously for all measures in the set X. :

$$R(X) \equiv P(Q(s) \geq Q(s')|x(s) \geq x(s')\ \forall x \in X)$$

According to this definition, we may not be able to predict the quality of any system output (i.e. a translation) with a highly *reliable* measure set, but we can ensure a system improvement when all measures corroborate the result. Then the *additive reliability* property can be stated as:

$$R(X \cup \{x\}) \geq R(X)$$

We could think of violating this property by adding, for instance, a measure consisting of a random function ($x'(s) = rand(0..1)$) or a reversal of the original measure ($x'(s) = 1/x(s)$). These kind of measures, however, would not satisfy the constraints defined in Section 3.

This property is based on the idea that similarity with human references according to any aspect should not imply statistically a quality decrease. Although our test suites includes measures with low correlation at segment and system level, we can confirm empirically that all of them satisfy this property.

We have developed the following experiment: taking all possible measure pairs in the test suites, we have compared their reliability as a set versus the maximal reliability of any of them (by computing the difference $R(X) - max(R(x_1), R(x_2))$. Figure 1 shows the obtained distribution of this difference for our MT and AS test suites. Remarkably, in almost every case this difference is positive.

This result has a key implication: Corroborating evaluation results with a new measure, even when it has lower correlation with human judgements, increases the reliability of results. Therefore, if the correlation with judgements is not determinant, the question is now what factor determines the contribution of the new measures. According to the following property, this factor is the heterogeneity of measures.

6 Heterogeneity

This property states that *the reliability of any measure combination is lower bounded by the heterogeneity of the measure set*. In other words, a single measure can be more or less reliable, but a system improvement according to all measures in an heterogeneous set is reliable.

Let us define the *heterogeneity* $H(X)$ of a set of measures X as, given two system outputs s and s' such that $g \neq s \neq s' \neq g$ (g is the reference text), the probability that there exist two measures that contradict each other. That is:

$$H(X) \equiv P(\exists x, x' \in X . x(s) > x(s') \wedge x'(s) < x'(s'))$$

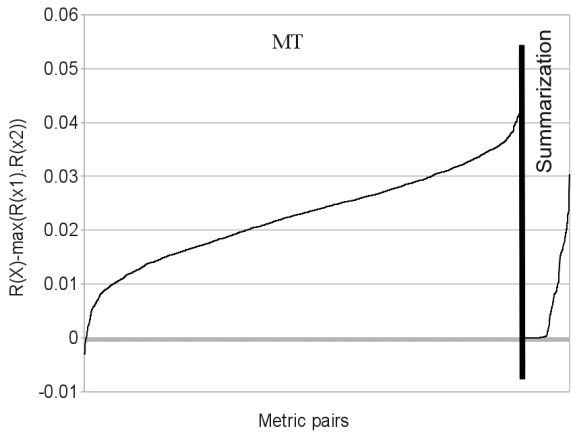


Figure 1: Additive reliability for metric pairs.

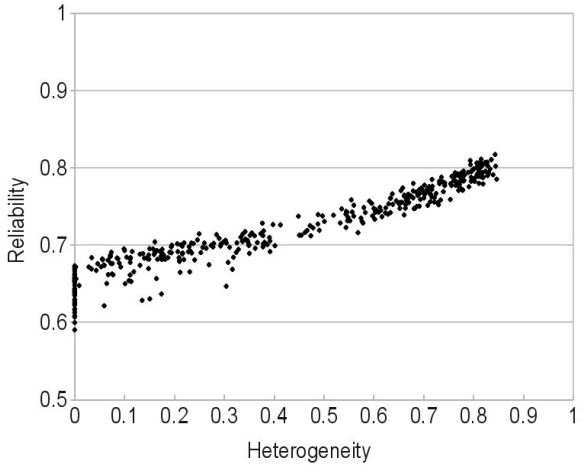


Figure 2: Heterogeneity vs. reliability in MT test suites.

Thus, given a set X of measures, the property states that there exists a strict growing function F such that:

$$R(X) \geq F(H(X)) \text{ and } H(X) = 1 \rightarrow R(X) = 1$$

In other words, the more the similarity measures tend to contradict each other, the more a unanimous improvement over all similarity measures is reliable. Clearly, the harder it is that measures agree, the more meaningful it is when they do.

The first part is derived from the *Additive Reliability* property. Intuitively, any individual measure has zero heterogeneity. Increasing the heterogeneity implies joining measures or measure sets progressively. According to the *Additive Reliability* property, this joining implies a reliability increase. Therefore, the higher the heterogeneity, the higher the minimum Reliability achieved by the corresponding measure sets.

The second part is derived from the *Heterogeneity* definition. If $H(X) = 1$ then, for any distinct pair of outputs that differ from the reference, there exist at least two measures in the set contradicting each other. That is, $H(X) = 1$ implies that:

$$\forall s \neq s' \neq g (\exists x, x' \in X . x(s) > x(s') \wedge x'(s) < x'(s'))$$

Therefore, if one output improves the other according to all measures, then the output must be equal than the reference.

$$\neg(\exists x, x' \in X . x(s) > x(s') \wedge x'(s) < x'(s')) \rightarrow$$

$$\neg(g \neq s \neq s' \neq g) \rightarrow g = s \vee g = s'$$

According to the first constraint of similarity measures, a text that is equal to the reference achieves the maximum score:

$$g = s \rightarrow f(g) = g(s) \rightarrow \forall x . x(s) \geq x(s')$$

Finally, if we assume that the reference (human produced texts) has a maximum quality, then it will have equal or higher quality than the other output.

$$g = s \rightarrow Q(s) \geq Q(s')$$

Therefore, the reliability of the measure set is maximal. In summary, if $H(X) = 1$ then:

$$R(X) = P(Q(s) \geq Q(s') | x(s) \geq x(s') \, \forall x \in X) =$$

$$= P(Q(s) \geq Q(s') | s = g) = 1$$

Figures 2 and 3 show the relationship between the heterogeneity of randomly selected measure sets and their reliability for the MT and summarization test suites. As the figures show, the higher the heterogeneity, the higher the reliability of the measure set. The results in AS are less pronounced due to the redundancy in ROUGE measure.

Notice that the heterogeneity property does not necessarily imply a high correlation between reliability and heterogeneity. For instance, an ideal single measure would have zero heterogeneity and

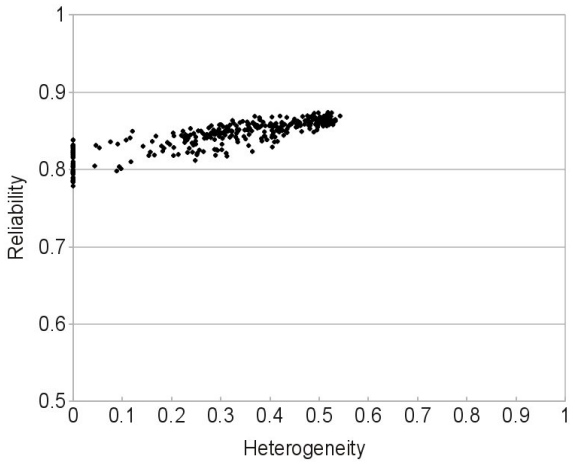


Figure 3: Heterogeneity vs. reliability in summarization test suites.

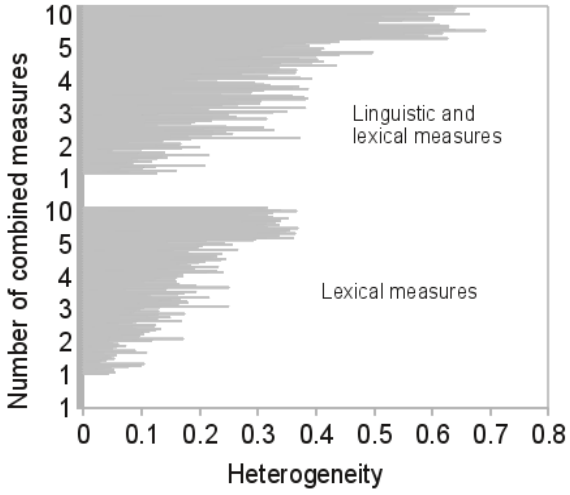


Figure 4: Heterogeneity of lexical measures vs. lexical and linguistic measures.

achieve maximum reliability, appearing in the top left area. The property rather brings us to the following situation: let us suppose that we have a set of single measures available which achieve a certain range of reliability. We can improve our system according to any of these measures. Without human assessments, we do not know what is the most reliable measure. But if we combine them, increasing the heterogeneity, the minimal reliability of the selected measures will be higher. This implies that combining heterogeneous measures (e.g. at high linguistic levels) that do not achieve high correlation in isolation, is better than corroborating results with any individual measure alone, such as ROUGE and BLEU, which is the common practice in the state of the art.

The main drawback of this property is that increasing the heterogeneity implies a sensitivity reduction. For instance, if $H(X) = 0.9$, then only for 10% of output pairs in the corpus there exists an improvement according to all measures. In other words, unanimous evaluation results from heterogeneous measures are reliable but harder to achieve for the system developer. The next section investigates on this issue.

Finally, Figure 4 shows that linguistic measures increase the heterogeneity of measure sets. We have generated sets of metrics of size 1 to 10 made up by lexical or lexical and linguistic metrics. As the figure shows, in the second case, the measure sets achieve a higher heterogeneity.

7 Score thresholds vs. Additive Reliability

According to the previous properties, corroborating evaluation results with several measures increases the reliability of evaluation results at the cost of sensitivity. On the other hand, increasing the score threshold of a single measure should have a similar effect. Which is then the best methodology to improve reliability? In this section we provide experimental evidence on the relationship between both ways of increasing reliability: we have found that, corroborating evaluation results over single texts with additional measures is more reliable than requiring higher score differences according to any individual measure in the set. More specifically, we have found that *the reliability of a measure set is higher than the reliability of each of the individual measures at a similar level of sensitivity.*

Formally, we define the sensitivity $S(X)$ of a metric set X as the probability of finding a score improvement within text pairs with a real (i.e. human assessed) quality improvement:

$$S(X) = P(x(s) \geq x(s')\forall x \in X | Q(s) \geq Q(s'))$$

Being $R_{th}(x)$ and $S_{th}(x)$ the reliability and sensitivity of a single measure x for a certain increase score threshold th:

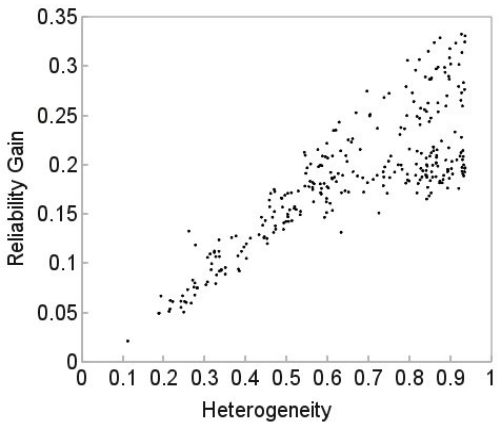


Figure 5: Heterogeneity vs. reliability Gain for MT test suites.

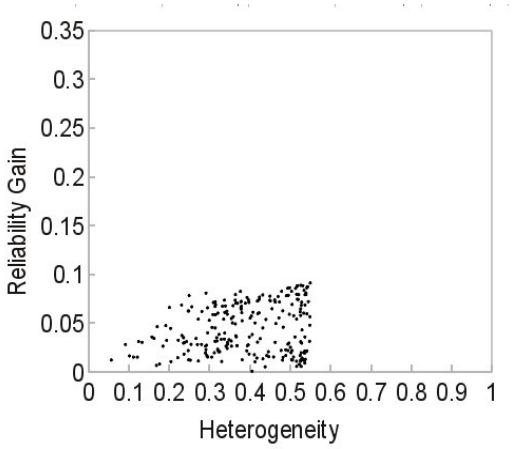


Figure 6: Heterogeneity vs. reliability Gain for MT test suites.

$$R_{th}(x) = P(Q(s) \geq Q(s')|x(s) - x(s') \geq th)$$
$$S_{th}(x) = P(x(s) - x(s') \geq th|Q(s) \geq Q(s'))$$

The property that we want to check is that, at the same sensitivity level, combining measures is more reliable than increasing the score threshold of single measures:

$$S(X) = S_{th}(x).x \in X \longrightarrow R(X) \geq R_{th}(x)$$

Note that if we had a perfect measure x_p such that $R(x_p) = S(x_p) = 1$, then combining this measure with a low reliability measure x_l would produce a lower sensitivity, but the maximal reliability would be preserved.

In order to confirm empirically this property, we have developed the following experiment: (i) We compute the reliability and sensitivity of randomly chosen measure sets over single text pairs. We have generated sets of 2,3,5,10,20 and 40 measures. In the case of summarization corpora we have combined up to 20 measures. In addition, we compute also the heterogeneity $H(X)$ of each measure set; (ii) Experimenting with different values for the threshold th, we compute the reliability of single measures for all potential sensitivity levels; (iii) For each measure set, we compare the reliability of the measure set versus the reliability of single measures at the same sensitivity level. We will refer to this as the *Reliability Gain*:

$$\text{Reliability Gain} =$$
$$R(X) - max\{R_{th}(x)/x \in X \wedge S_{th}(x) = S(X)\}$$

If there are several reliability values with the same sensitivity for a given single measures, we choose the highest reliability value for the single measure.

Figures 5 and 6 illustrate the results for the MT and AS corpora. The horizontal axis represents the Heterogeneity of measure sets, while the vertical axis represents the reliability gain. Remarkably, the reliability gain is positive for all cases in our test suites. The maximum reliability gain is 0.34 in the case of MT and 0.08 for AS (note that summarization measures are more redundant in our corpora). In both test suites, the largest information gains are obtained with highly heterogeneous measure sets.

In summary, given comparable measures in terms of reliability, corroborating evaluation results with several measures is more effective than optimizing systems according to the best measure in the set. This empirical property provides an additional evidence in favour of the use of heterogeneous measures and, in particular, of the use of linguistic measures in combination with standard lexical measures.

8 Conclusions

In this paper, we have analyzed the state of the art in order to clarify why novel text evaluation measures

are not exploited by the community. Our first conclusion is that it is not easy to determine the reliability of measures, which is highly corpus-dependent and often contradictory when comparing correlation with human judgements at segment vs. system levels.

In order to tackle this issue, we have studied a number of properties that suggest the convenience of using heterogeneous measures to corroborate evaluation results. According to these properties, we can ensure that, even when if we can not determine the reliability of individual measures, corroborating a system improvement with additional measures always increases the reliability of the results. In addition, the more heterogeneous the measures employed (which is measurable), the higher the reliability of the results. But perhaps the most important practical finding is that the reliability at similar sensitivity levels by corroborating evaluation results with several measures is always higher than improving systems according to any of the combined measures in isolation.

These properties point to the practical advantages of considering linguistic knowledge (beyond lexical information) in measures, even if they do not achieve a high correlation with human judgements. Our experiments show that linguistic knowledge increases the heterogeneity of measure sets, which in turn increases the reliability of evaluation results when corroborating system comparisons with several measures.

Acknowledgements

This work has been partially funded by the Spanish Government (Holopedia, TIN2010-21128-C02 and OpenMT-2, TIN2009-14675-C03) and the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 247762 (FAUST project, FP7-ICT-2009-4-247762).

References

Yasuhiro Akiba, Kenji Imamura, and Eiichiro Sumita. 2001. Using Multiple Edit Distances to Automatically Rank Machine Translation Output. In *Proceedings of Machine Translation Summit VIII*, pages 15–20.

Joshua Albrecht and Rebecca Hwa. 2007a. A Re-examination of Machine Learning Approaches for Sentence-Level MT Evaluation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 880–887.

Joshua Albrecht and Rebecca Hwa. 2007b. Regression for Sentence-Level MT Evaluation with Pseudo References. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 296–303.

Enrique Amigó, Julio Gonzalo, Anselmo Pe nas, and Felisa Verdejo. 2005. QARLA: a Framework for the Evaluation of Automatic Summarization. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 280–289.

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Lluís Màrquez. 2006. MT Evaluation: Human-Like vs. Human Acceptable. In *Proceedings of the Joint 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 17–24.

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Felisa Verdejo. 2009. The contribution of linguistic features to automatic machine translation evaluation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 306–314, Stroudsburg, PA, USA. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

Chris Callison-burch and Miles Osborne. 2006. Re-evaluating the role of bleu in machine translation research. In *In EACL*, pages 249–256.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 workshop on statistical machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53. Revised August 2010.

Yee Seng Chan and Hwee Tou Ng. 2008. MAXSIM: A maximum similarity metric for machine translation evaluation. In *Proceedings of ACL-08: HLT*, pages 55–62.

Simon Corston-Oliver, Michael Gamon, and Chris Brockett. 2001. A Machine Learning Approach to the Automatic Evaluation of Machine Translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 140–147.

Christopher Culy and Susanne Z. Riehemann. 2003. The Limits of N-gram Translation Evaluation Metrics. In *Proceedings of MT-SUMMIT IX*, pages 1–8.

George Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. In *Proceedings of the 2nd International Conference on Human Language Technology*, pages 138–145.

Michael Gamon, Anthony Aue, and Martine Smets. 2005. Sentence-Level MT evaluation without reference translations: beyond language modeling. In *Proceedings of EAMT*, pages 103–111.

Jesús Giménez and Lluís Màrquez. 2007. Linguistic Features for Automatic Evaluation of Heterogeneous MT Systems. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 256–264.

Jesús Giménez and Lluís Màrquez. 2008. Heterogeneous Automatic MT Evaluation Through Non-Parametric Metric Combinations. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, pages 319–326.

Jesús Giménez and Lluís Màrquez. 2010. Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation. *The Prague Bulletin of Mathematical Linguistics*, 1(94):77–86.

Jesús Giménez. 2008. *Empirical Machine Translation and its Evaluation*. Ph.D. thesis, Universitat Politècnica de Catalunya.

Tsutomu Hirao, Manabu Okumura, and Hideki Isozaki. 2005. Kernel-based approach for automatic evaluation of natural language generation technologies: Application to automatic summarization. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 145–152, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Petr Homola, Vladislav Kuboň, and Pavel Pecina. 2009. A simple automatic mt evaluation metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 33–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeremy G. Kahn, Matthew Snover, and Mari Ostendorf. 2009. Expected Dependency Pair Match: Predicting translation quality with expected syntactic structure. *Machine Translation*.

Alex Kulesza and Stuart M. Shieber. 2004. A learning approach to improving sentence-level MT evaluation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 75–84.

LDC. 2005. Linguistic Data Annotation Specification: Assessment of Adequacy and Fluency in Translations. Revision 1.5. Technical report, Linguistic Data Consortium. `http://www.ldc.upenn.edu/Projects/TIDES/Translation/TransAssess04.pdf`.

Audrey Le and Mark Przybocki. 2005. NIST 2005 machine translation evaluation official results. In *Official release of automatic evaluation scores for all submissions, August*.

Gregor Leusch, Nicola Ueffing, and Hermann Ney. 2006. CDER: Efficient MT Evaluation Using Block Movements. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 241–248.

Chin-Yew Lin and Franz Josef Och. 2004a. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chin-Yew Lin and Franz Josef Och. 2004b. ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*.

Chin-Yew Lin. 2004. Rouge: A Package for Automatic Evaluation of Summaries. In Marie-Francine Moens and Stan Szpakowicz, editors, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Lucian Vlad Lita, Monica Rogati, and Alon Lavie. 2005. BLANC: Learning Evaluation Metrics for MT. In *Proceedings of the Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 740–747.

Ding Liu and Daniel Gildea. 2005. Syntactic Features for Evaluation of Machine Translation. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 25–32.

Ding Liu and Daniel Gildea. 2006. Stochastic Iterative Alignment for Machine Translation Evaluation. In *Proceedings of the Joint 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 539–546.

Ding Liu and Daniel Gildea. 2007. Source-Language Features and Maximum Correlation Training for Machine Translation Evaluation. In *Proceedings of the 2007 Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 41–48.

Dennis Mehay and Chris Brew. 2007. BLEUATRE: Flattening Syntactic Dependencies for MT Evaluation. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. 2003. Precision and Recall of Machine Translation. In *Proceedings of the Joint Conference on Human Language Technology and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*.

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2007a. Dependency-Based Automatic Evaluation for Machine Translation. In *Proceedings of SSST, NAACL-HLT/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 80–87.

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2007b. Labelled Dependencies in Machine Translation Evaluation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 104–111.

Karolina Owczarzak, Josef van Genabith, and Andy Way. 2008. Evaluating machine translation with lfg dependencies. *Machine Translation*, 21(2):95–119.

Karolina Owczarzak. 2009. Depeval(summ): dependency-based evaluation for automatic summaries. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 190–198, Morristown, NJ, USA. Association for Computational Linguistics.

Sebastian Padó, Michael Galley, Dan Jurafsky, and Christopher D. Manning. 2009. Robust machine translation evaluation with entailment features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 297–305.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001a. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, jul.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001b. Bleu: a method for automatic evaluation of machine translation, RC22176. Technical report, IBM T.J. Watson Research Center.

Michael Paul, Andrew Finch, and Eiichiro Sumita. 2007. Reducing Human Assessments of Machine Translation Quality to Binary Classifiers. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*.

Maja Popovic and Hermann Ney. 2007. Word Error Rates: Decomposition over POS classes and Applications for Error Analysis. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 48–55, Prague, Czech Republic, June. Association for Computational Linguistics.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–279.

Chris Quirk. 2004. Training a Sentence-Level Machine Translation Confidence Metric. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 825–828.

Florence Reeder, Keith Miller, Jennifer Doyon, and John White. 2001. The Naming of Things and the Confusion of Tongues: an MT Metric. In *Proceedings of the Workshop on MT Evaluation "Who did what to whom?" at Machine Translation Summit VIII*, pages 55–59.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 223–231.

Christoph Tillmann, Stefan Vogel, Hermann Ney, A. Zubiaga, and H. Sawaf. 1997. Accelerated DP based Search for Statistical Translation. In *Proceedings of European Conference on Speech Communication and Technology*.

Stephen Tratz and Eduard Hovy. 2008. Summarization evaluation using transformed basic elements. In *In Proceedings of TAC-08. Gaithersburg, Maryland*.

Joseph Turian, Luke Shen, and I. Dan Melamed. 2003a. Evaluation of machine translation and its evaluation. In *In Proceedings of MT Summit IX*, pages 386–393.

Joseph P. Turian, Luke Shen, and I. Dan Melamed. 2003b. Evaluation of Machine Translation and its Evaluation. In *Proceedings of MT SUMMIT IX*.

Ranking Human and Machine Summarization Systems

Peter Rankel
University of Maryland
College Park, Maryland
rankel@math.umd.edu

John M. Conroy
IDA/Center for Computing Sciences
Bowie, Maryland
conroyjohnm@gmail.com

Eric V. Slud
University of Maryland
College Park, Maryland
evs@math.umd.edu

Dianne P. O'Leary
University of Maryland
College Park, Maryland
oleary@cs.umd.edu

Abstract

The Text Analysis Conference (TAC) ranks summarization systems by their average score over a collection of document sets. We investigate the statistical appropriateness of this score and propose an alternative that better distinguishes between human and machine evaluation systems.

1 Introduction

For the past several years, the National Institute of Standards and Technology (NIST) has hosted the Text Analysis Conference (TAC) (previously called the Document Understanding Conference (DUC)) (Nat, 2010). A major theme of this conference is multi-document summarization: machine summarization of sets of related documents, sometimes query-focused and sometimes generic. The summarizers are judged by how well the summaries match human-generated summaries in either automatic metrics such as ROUGE (Lin and Hovy, 2003) or manual metrics such as responsiveness or pyramid evaluation (Nenkova et al., 2007). Typically the systems are ranked by their average score over all document sets.

Ranking by average score is quite appropriate under certain statistical hypotheses, for example, when each sample is drawn from a distribution which differs from the distribution of other samples only through a location shift (Randles and Wolfe, 1979). However, a non-parametric (rank-based) analysis of variance on the summarizers' scores on each document set revealed an impossibly small p-value (less

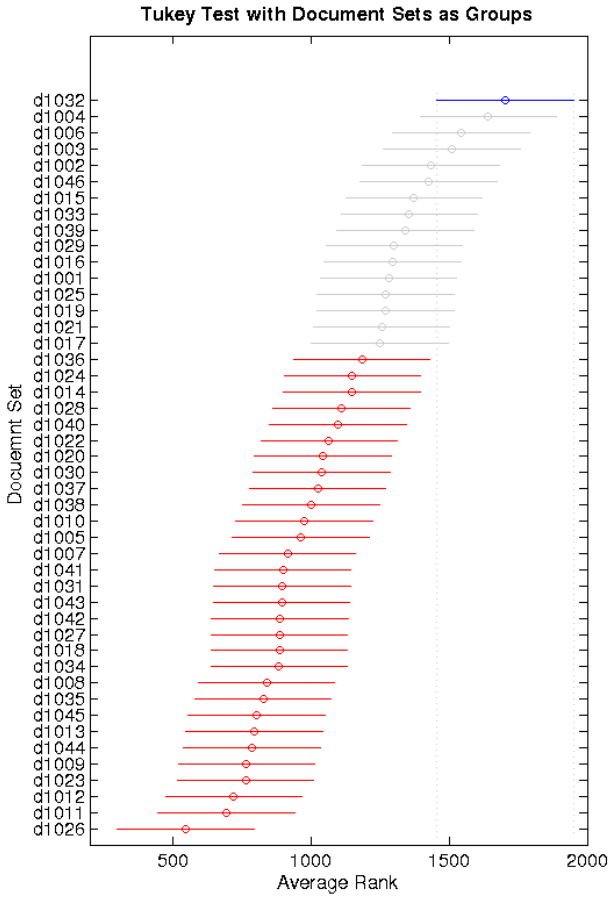


Figure 1: Confidence Intervals from a non-parametric Tukey's honestly significant difference test for 46 TAC 2010 update document sets. The blue confidence interval (for document set d1032) does not overlap any of the 30 red intervals. Hence, the test concludes that 30 document sets have mean significantly different from the mean of d1032.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 467–473,
Edinburgh, Scotland, UK, July 27–31, 2011.

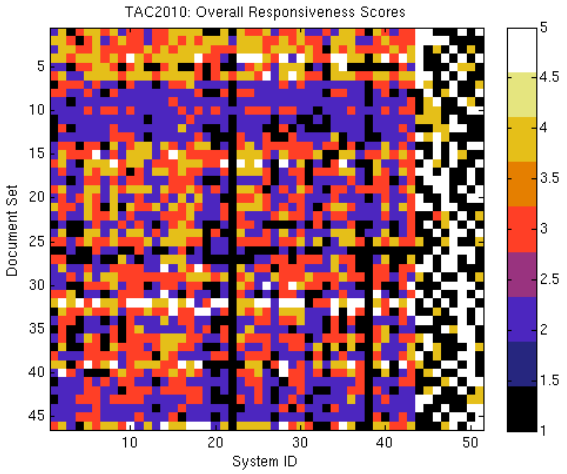


Figure 2: Overall Responsiveness scores.

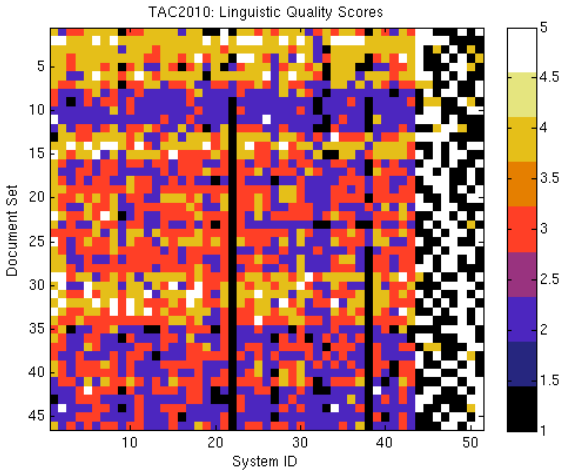


Figure 3: Linguistic scores.

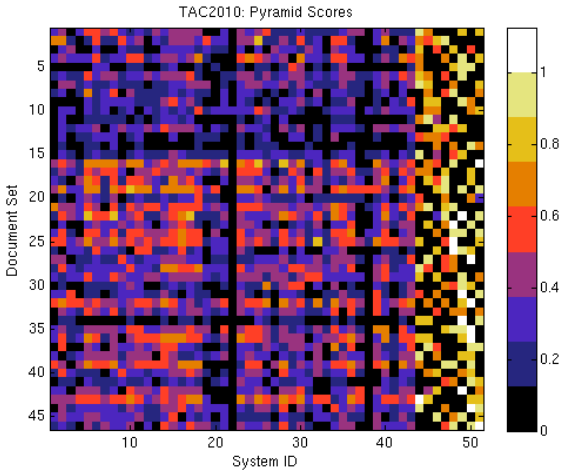


Figure 4: Pyramid scores.

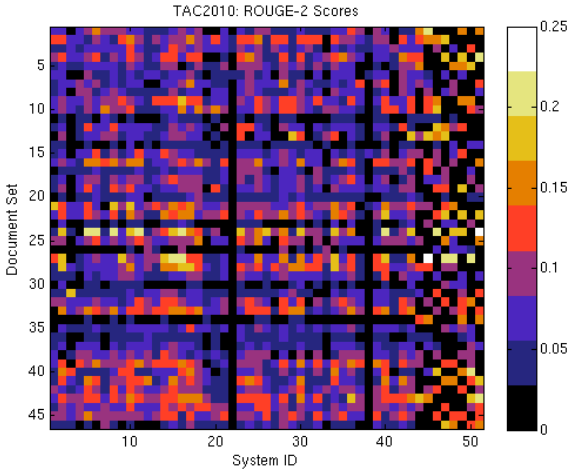


Figure 5: ROUGE-2 scores for the TAC 2010 update summary task, organized by document set (y-axis) and summarizer (x-axis). The 51 summarizers fall into two distinct groups: machine systems (first 43 columns) and humans (last 8 columns). Note that each human only summarized half of the document sets, thus creating 23 missing values in each of the last 8 columns. Black is used to indicate missing values in the last 8 columns and low scores in the first 43 columns.

than 10^{-12} using Matlab's `kruskalwallis` [1]), providing evidence that a summary's score is not independent of the document set. This effect can be seen in Figure 1, showing the confidence bands, as computed by a Tukey honestly significant difference test for each document set's difficulty as measured by the mean rank responsiveness score for TAC 2010. The test clearly shows that the summarizer performances on different document sets have different averages.

We further illustrate this in Figures 2 – 5, which show the scores of various summarizers on various document sets using standard human and automatic evaluation methods (Dang and Owczarzak, 2008) of overall responsiveness, linguistic quality, pyramid scores, and ROUGE-2 using color to indicate the value of the score. Some rows are clearly darker, indicating overall lower scores for the sum-

[1]The Kruskal-Wallis test performs a one-way analysis of variance of document-set differences after first converting the summary scores for each sample to their ranks within the pooled sample. Computed from the converted scores, the Kruskal-Wallis test statistic is essentially the ratio of the between-group sum of squares to the combined within-group sum of squares.

maries of these documents, and the variances of the scores differ row-by-row. These plots show qualitatively what the non-parametric analysis of variance demonstrates statistically. While the data presented was for the TAC 2010 update document sets, similar results hold for all the TAC 2008, 2009, and 2010 data. Hence, it may be advantageous to measure summarizer quality by accounting for heterogeneity of documents within each test set. A non-parametric paired test like the Wilcoxon signed-rank is one way to do this. Another way would be paired t-tests.

In the paper (Conroy and Dang, 2008) the authors noted that while there is a significant gap in performance between machine systems and human summarizers when measured by average manual metrics, this gap is not present when measured by the averages of the best automatic metric (ROUGE). In particular, in the DUC 2005-2007 data some systems have ROUGE performance within the 95% confidence intervals of several human summarizers, but their pyramid, linguistic, and responsiveness scores do not achieve this level of performance. Thus, the inexpensive automatic metrics, as currently employed, do not predict well how machine summaries compare to human summaries.

In this work we explore the use of document-paired testing for summarizer comparison. Our main approach is to consider each pair of two summarizers' sets of scores (over all documents) as a balanced two-sample dataset, and to assess that pair's mean difference in scores through a two-sample T or Wilcoxon test, paired or unpaired. Our goal has been to confirm that human summarizer scores are uniformly different and better on average than machine summarizer scores, and to rate the quality of the statistical method (T or W, paired or unpaired) by the consistency with which the human versus machine scores show superior human performance. Our hope is that paired testing, using either the standard paired two-sample t-test or the distribution-free Wilcoxon signed-rank test, can provide greater power in the statistical analysis of automatic metrics such as ROUGE.

2 Size and Power of Tests

Statistical tests are generally compared by choosing rejection thresholds to achieve a certain small probability of Type I error (usually as $\alpha = .05$). Given multiple tests with the same Type I error, one prefers the test with the smallest probability of Type II error. Since power is defined to be one minus the Type II error probability, we prefer the test with the most power. Recall that a *test-statistic* S depending on available data-samples gives rise to a *rejection region* by defining rejection of the null hypothesis H_0 as the event $\{S \geq c\}$ for a *cutoff* or *rejection threshold* c chosen so that

$$P(S \geq c) \leq \alpha$$

for all probability laws compatible with the null hypothesis where the (nominal) *significance level* α is chosen in advance by the statistician, usually as $\alpha = .05$. However, in many settings, the null hypothesis comprises many possible probability laws, as here where the null hypothesis is that the underlying probability laws for the score-samples of two separate summarizers are equal, without specifying exactly what that probability distribution is. In this case, the significance level is an upper bound for the attained *size* of the test, defined as $\sup_{P \in H_0} P(S \geq c)$, the largest rejection probability $P(S \geq c)$ achieved by any probability law compatible with the null hypothesis. The power of the test then depends on the specific probability law Q from the considered alternatives in H_A. For each such Q, and given a threshold c, the power for the test at Q is the rejection probability $Q(S \geq c)$. These definitions reflect the fact that the null and alternative hypotheses are *composite*, that is, each consists of multiple probability laws for the data. One of the advantages of considering a *distribution-free* two-sample test statistic such as the Wilcoxon is that the probability distribution for the statistic S is then the same for all (continuous, or non-discrete) probability laws $P \in H_0$, so that one cutoff c serves for all of H_0 with all rejection probabilities equal to α. [2]

Two test statistics, say S and $\tilde{S}$, are generally compared in terms of their powers at fixed alternatives Q in the alternative hypothesis H_A, when their respective thresholds c, c^* have been defined so that the sizes of the respective tests, $\sup_{P \in H_0} P(S \geq$

[2] The Wilcoxon test is not distribution-free for discrete data. However, the discrete TAC data can be thought of as rounded continuous data, rather than as truly discrete data.

c) and $\sup_{P \in H_0} P(\tilde{S} \geq c^*)$, are approximately equal. In this paper, the test statistics under consideration are – in one-sided testing — the (unpaired) two-sample t test with pooled sample variance (T), the paired two-sample t test (T^p), and the (paired) signed-rank Wilcoxon test (W); and for two-sided testing, S is defined by the absolute value of one of these statistics. The thresholds c for the tests can be defined either by theoretical distributions, by large-sample approximations, or by data-resampling (*bootstrap*) techniques, and (only) in the last case are these thresholds data-dependent, or random. We explain these notions with respect to the two-sample data-structure in which the scores from the first summarizer are denoted $X_1, \ldots, X_n$, where n is the number of documents with non-missing scores for both summarizers, and the scores from the second summarizer are $Y_1, \ldots, Y_n$. Let $Z_k = X_k - Y_k$ denote the document-wise differences between the summarizers' scores, and $\bar{Z} = n^{-1} \sum_{k=1}^{n} Z_k$ be their average. Then the paired statistics are defined as

$$T^p = \sqrt{n(n-1)}\, \bar{Z} / (\sum_{k=1}^{n} (Z_k - \bar{Z})^2)^{1/2}$$

and

$$W = \sum_{k=1}^{n} \mathrm{sgn}(Z_k)\, R_k^+$$

where R_k^+ is the rank of $|Z_k|$ among $|Z_1|, \ldots, |Z_n|$. Note that under both null and alternative hypotheses, the variates Z_k are assumed independent identically distributed (*iid*), while under H_0, the random variables Z_k are symmetric about 0.

The t-statistic T^p is 'parametric' in the sense that exact theoretical calculations of probabilities $P(a < T^p < b)$ depend on the assumption of normality of the differences Z_k, and when that holds, the two-sided cutoff $c = c(T^p)$ is defined as the $1 - \alpha/2$ quantile of the t_{n-1} distribution with $n - 1$ degrees of freedom. However, when n is moderately or very large, the cutoff is well approximated by the standard-normal $1 - \alpha/2$ quantile $z_{\alpha/2}$, and T^p becomes approximately nonparametrically valid with this cutoff, by the Central Limit Theorem. The Wilcoxon signed-rank statistic W has theoretical cutoff $c = c(W)$ which depends only on n, whenever the data Z_k are continuously distributed; but for large n, the cutoff is given simply as $\sqrt{n^3/12} \cdot z_{\alpha/2}$. When there are ties (as might be common in discrete data), the calculation of cutoffs and p-values for Wilcoxon becomes slightly more complicated and is no longer fully nonparametric except in a large-sample approximate sense.

The situation for the two-sample unpaired t-statistic T currently used in TAC evaluation is not so neat. Even when the two samples $\mathbf{X} = \{X_k\}_{k=1}^n$ and $\mathbf{Y} = \{Y_k\}_{k=1}^n$ are independent, exact theoretical distribution of cutoffs is known only under the parametric assumption that the scores are normally distributed (and in the case of the pooled-sample-variance statistic, that $\mathrm{Var}(X_k) = \mathrm{Var}(Y_k)$.) However, an essential element of the summarization data is the heterogeneity of documents. This means that while $\{X_k\}_{k=1}^n$ can be viewed as *iid* scores when documents are selected randomly – and not necessarily equiprobably – from the ensemble of all possible documents, the Y_k and X_k samples are *dependent*. Still, the pairs $\{(X_k, Y_k)\}_{k=1}^n$, and therefore the differences $\{Z_k\}_{k=1}^n$, are *iid* which is what makes paired testing valid. However, there is no theoretical distribution for T from which to calculate valid quantiles c for cutoffs, and therefore the use of the unpaired t-statistic cannot be recommended for TAC evaluation.

What can be done in a particular dataset, like the TAC summarization score datsets we consider, to ascertain the approximate validity of theoretically derived large-sample cutoffs for test statistics? In the age of plentiful and fast computers, quite a lot, through the powerful computational machinery of the *bootstrap* (Efron and Tibshirani, 1993).

The idea of bootstrap hypothesis testing (Efron and Tibshirani, 1993), (Bickel and Ren, 2001) is to randomly sample with replacement (the rows with non-missing data in) the dataset $\{(X_k, Y_k)\}_{k=1}^n$ in such a way as to generate representative data that plausibly *would* have been seen if two-sample score data had been generated from two equally effective summarizers with score distributional characteristics like the pooled scores from the two observed summarizers. We have done this in two distinct ways, each creating 2000 datasets with n paired scores:

MC *Monte Carlo Method.* For each of many it-

erations (in our case 2000), define a new dataset $\{(X'_k, Y'_k)\}_{k=1}^n$ by independently swapping X_k and Y_k with probability $1/2$. Hence, $(X'_k, Y'_k) = (X_k, Y_k)$ with probability $1/2$ and (Y_k, X_k) with probability $1/2$.

HB *Hybrid MC/Bootstrap.* For each of 2000 iterations, create a re-sampled dataset $\{(X''_k, Y''_k)\}_{k=1}^n$ in the following way. First, sample n pairs (X_k, Y_k) with replacement from the original dataset. Then, as above, randomly swap the components of each pair, each with $1/2$ probability.

Both of these two methods can be seen to generate two-sample data satisfying H_0, with each score-sample's distribution obtained as a mixture of the distributions actually generating the **X** and **Y** samples. The *empirical* q^{th} *quantiles* for a statistic $S = S(\mathbf{X}, \mathbf{Y})$ such as $|W|$ or $|T^p|$ are estimated from the resampled data as $\hat{F}_S^{-1}(q)$, where $\hat{F}_S(t)$ is simply the fraction of times (out of 2000) that the statistic S applied to the constructed dataset had a value less than or equal to t. The upshot is that the $1-\alpha$ empirical quantile for S based on either of these simulation methods serves as a data-dependent cutoff c attaining approximate size α for all H_0-generated data. The MC and HB methods will be employed in Section 4 to check the theoretical p-values.

3 Relative Efficiency of W versus T^p

Statistical theory does have something to say about the comparative powers of paired W versus T^p statistics. These statistics have been studied (Randles and Wolfe, 1979), in terms of their *asymptotic relative efficiency* for location-shift alternatives based on symmetric densities ($f(z-\vartheta)$ is a location-shift of $f(z)$). For many pairs of parametric and rank-based statistics $S, \tilde{S}$, including W and T^p, the following assertion has been proved for testing H_0 at significance level α.

First assume the Z_k are distributed according to some density $f(z-\vartheta)$, where $f(z)$ is a symmetric function ($f(-z) = f(z)$). Next assume $\vartheta = 0$ under H_0. When n gets large the powers at any alternatives with very small $\vartheta = \gamma/\sqrt{n},\ \gamma \neq 0$, can be made asymptotically equal by using samples of size n with statistic S and of size $\rho \cdot n$ with statistic $\tilde{S}$. Here $\rho = ARE(S, \tilde{S})$ is a constant not depending on n or γ but definitely depending on f, called *asymptotic relative efficiency* of S with respect to $\tilde{S}$. (The smaller $\rho < 1$ is, the more statistic $\tilde{S}$ is preferred among the two.)

Using this definition, it is known (Randles and Wolfe 1979, Sec. 5.4 leading up to Table 5.4.7 on p. 167) that the Wilcoxon signed-rank statistic W provides greater robustness and often much greater efficiency than the paired T, with ARE which is 0.95 with f a standard normal density, and which is never less than 0.864 for any symmmetric density f. However, in our context, continuous scores such as pyramid exhibit document-specific score differences between summarizers which often have approximately normal-looking histograms, and although the alternatives perhaps cannot be viewed as pure location shifts, it is unsurprising in view of the ARE theory cited above that the W and T paired tests have very similar performance. Nevertheless, as we found by statistical analysis of the TAC data, both are far superior to the unpaired T-statistic, with either theoretical or empirical bootstrapped p-values.

4 Testing Setup and Results

To evaluate our ideas, we used the TAC data from 2008-2010 and focused on three manual metrics (overall responsiveness, pyramid score, and linguistic quality score) and two automatic metrics (ROUGE-2 and ROUGE-SU4). We make the assumption, backed by both the scores given and comments made by NIST summary assessors [3], that automatic summarization systems do not perform at the human level of performance. As such, if a statistic based on an automatic metric, such as ROUGE-2, were to show fewer systems performing at human level of performance than the statistic of averaging scores, such a statistic would be preferable because

[3] Assessors have commented privately at the Text Analysis Conference 2008, that while the origin of the summary is hidden from them, "we know which ones are machine generated." Thus, automatic summarization fails the Turing test of machine intelligence (Turing, 1950). This belief is also supported by (Conroy and Dang, 2008) and (Dang and Owczarzak, 2008). Finally, our own results show no matter how you compare human and machine scores all machines systems score significantly worse than humans.

	2008: $2145 = \binom{66}{2}$ pairs			2009: $1830 = \binom{61}{2}$ pairs			2010: $1275 = \binom{51}{2}$ pairs		
Metric	**Unpair-T**	**Pair-T**	**Wilc.**	**Unpair-T**	**Pair-T**	**Wilc.**	**Unpair-T**	**Pair-T**	**Wilc.**
Linguistic	1234	**1416**	1410	1000	**1182**	1173	841	**939**	934
Overall	1202	**1353**	1342	982	**1149**	1146	845	**894**	889
Pyramid	1263	1417	**1418**	1075	**1238**	1216	875	**933**	926
ROUGE-2	1243	1453	**1459**	1016	1182	**1193**	812	938	**939**
ROUGE-SU4	1333	1493	**1507**	1059	1241	**1254**	894	**983**	976

Table 1: Number of significant differences found when testing for the difference of all pairs of summarization systems (including humans).

	2008: $464 = 58 \times 8$ pairs			2009: $424 = 53 \times 8$ pairs			2010: $344 = 43 \times 8$ pairs		
Metric	**Unpair-T**	**Pair-T**	**Wilc.**	**Unpair-T**	**Pair-T**	**Wilc.**	**Unpair-T**	**Pair-T**	**Wilc.**
Linguistic	464	464	464	424	424	424	344	344	344
Overall	464	464	464	424	424	424	344	344	344
Pyramid	464	464	464	424	424	424	344	344	344
ROUGE-2	375	**409**	402	323	**350**	341	275	**309**	305
ROUGE-SU4	391	**418**	414	354	**378**	373	324	**331**	328

Table 2: Number of significant differences resulting from $8 \times (N - 8)$ tests for human-machine system means or signed-rank comparisons.

of its greater power in the machine vs. human summarization domain.

For each of these metrics, we first created a score matrix whose (i, j)-entry represents the score for summarizer j on document set i (these matrices generated the colorplots in Figures 2 – 5). We then performed a Wilcoxon signed-rank test on certain pairs of columns of this matrix (any pair consisting of one machine system and one human summarizer). As a baseline, we did the same testing with a paired and an unpaired t-test. Each of these tests resulted in a p-value, and we counted how many were less than .05 and called these the significant differences.

The results of these tests (shown in Table 2), were somewhat surprising. Although we expected the nonparametric signed-rank test to perform better than an unpaired t-test, we were surprised to see that a paired t-test performed even better. All three tests always reject the null hypotheses when human metrics are used. This is what we'd like to happen with automatic metrics as well. As seen from the table, the paired t-test and Wilcoxon signed-rank test offer a good improvement over the unpaired t-test.

The results in Table 1 are less clear, but still positive. In this case, we are comparing pairs of machine summarization systems. In contrast to the human vs. machine case, we do not know the truth here. However, since the number of significant differences increases with paired testing here as well, we believe this also reflects the greater discriminatory power of paired testing.

We now apply the Monte Carlo and Hybrid Monte Carlo to check the theoretical p-values reported in Tables 1 and 2. The empirical quantiles found by these methods generally confirm the theoretical p-value test results reported there, especially in Table 2. In the overall tallies of all comparisons (Table 1), it seems that the bootstrap results (comparing only W and the un-paired T) make W look still stronger for linguistic and overall responsiveness versus the T; but for the pyramid and ROUGE scores, the bootstrap p-values bring T slightly closer to W although it still remains clearly inferior, achieving roughly 10% fewer rejections.

5 Conclusions and Future Work

In this paper we observed that summarization systems' performance varied significantly across document sets on the Text Analysis Conference (TAC) data. This variance in performance suggested that paired testing may be more appropriate than the t-test currently employed at TAC to compare the

performance of summarization systems. We proposed a non-parametric test, the Wilcoxon signed-rank test, as a robust more powerful alternative to the t-test. We estimated the statistical power of the t-test and the Wilcoxon signed-rank test by calculating the number of machine systems whose performance was significantly different than that of human summarizers. As human assessors score machine systems as not achieving human performance in either content or responsiveness, automatic metrics such as ROUGE should ideally indicate this distinction. We found that the paired Wilcoxon test significantly increases the number of machine systems that score significantly different than humans when the pairwise test is performed on ROUGE-2 and ROUGE-SU4 scores. Thus, we demonstrated that the Wilcoxon paired test shows more statistical power than the t-test for comparing summarization systems.

Consequently, the use of paired testing should not only be used in formal evaluations such as TAC, but also should be employed by summarization developers to more accurately assess whether changes to an automatic system give rise to improved performance.

Further study needs to analyze more summarization metrics such as those proposed at the recent NIST evaluation of automatic metrics, Automatically Evaluating Summaries of Peers (AESOP) (Nat, 2010). As metrics become more sophisticated and aim to more accurately predict human judgements such as overall responsiveness and linguistic quality, paired testing seems likely to be a more powerful statistical procedure than the unpaired t-test for head-to-head summarizer comparisons.

Throughout our research in this paper, we treated each separate kind of scores on a document set as data for one summarizer to be compared with the same kind of scores for other summarizers. However, it might be more fruitful to treat *all* the scores as multivariate data and compare the summarizers that way. Multivariate statistical techniques such as Principal Component Analysis may play a constructive role in suggesting highly discriminating new composite scores, perhaps leading to statistics with even more power to measure a summary's quality.

ROUGE was inspired by the success of the BLEU (BiLingual Evaluation Understudy), an n-gram based evaluation for machine translation (Papineni et al., 2002). It is likely that paired testing may also be appropriate for BLEU as well and will give additional discriminating power between machine translations and human translations.

References

Peter J. Bickel and Jian-Jian Ren. 2001. The Bootstrap in Hypothesis Testing. In *State of the Art in Statistics and Probability Theory, Festschrift for Willem R. van Zwet*, volume 36 of *Lecture Notes– Monograph Series*, pages 91–112. Institute of Mathematical Statistics.

John M. Conroy and Hoa Trang Dang. 2008. Mind the Gap: Dangers of Divorcing Evaluations of Summary Content from Linguistic Quality. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 145–152, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hoa T. Dang and Karolina Owczarzak. 2008. Overview of the tac 2008 update summarization task. In *Proceedings of the 1st Text Analysis Conference (TAC)*, Gaithersburg, Maryland, USA.

B. Efron and R. J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-Occurrences Statistics. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, Edmonton, Alberta.

National Institute of Standards and Technology. 2010. *Text Analysis Conference, http://www.nist.gov/tac*.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The Pyramid Method: Incorporating Human Content Selection Variation in Summarization Evaluation. *ACM Transactions on Speech and Language Processing*, 4(2).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

R.H. Randles and D.A. Wolfe. 1979. *Introduction to the Theory of Nonparametric Statistics*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley.

Alan Turing. 1950. Computing Machinery and Intelligence. *Mind*, 59(236):433–460.

Quasi-Synchronous Phrase Dependency Grammars for Machine Translation

Kevin Gimpel Noah A. Smith
Language Technologies Institute
Carnegie Mellon Univeristy
Pittsburgh, PA 15213, USA
{kgimpel,nasmith}@cs.cmu.edu

Abstract

We present a quasi-synchronous dependency grammar (Smith and Eisner, 2006) for machine translation in which the leaves of the tree are *phrases* rather than words as in previous work (Gimpel and Smith, 2009). This formulation allows us to combine structural components of phrase-based and syntax-based MT in a single model. We describe a method of extracting phrase dependencies from parallel text using a target-side dependency parser. For decoding, we describe a coarse-to-fine approach based on lattice dependency parsing of phrase lattices. We demonstrate performance improvements for Chinese-English and Urdu-English translation over a phrase-based baseline. We also investigate the use of *unsupervised* dependency parsers, reporting encouraging preliminary results.

1 Introduction

Two approaches currently dominate statistical machine translation (MT) research. Phrase-based models (Koehn et al., 2003) excel at capturing local reordering phenomena and memorizing multi-word translations. Models that employ syntax or syntax-like representations (Chiang, 2005; Galley et al., 2006; Zollmann and Venugopal, 2006; Huang et al., 2006) handle long-distance reordering better than phrase-based systems (Auli et al., 2009) but often require constraints on the formalism or rule extraction method in order to achieve computational tractability. As a result, certain instances of syntactic divergence are more naturally handled by phrase-based systems (DeNeefe et al., 2007).

In this paper we present a new way of combining the advantages of phrase-based and syntax-based MT. We propose a model in which phrases are organized into a tree structure inspired by dependency syntax. Instead of standard dependency trees in which *words* are vertices, our trees have *phrases* as vertices. We describe a simple heuristic to extract phrase dependencies from an aligned parallel corpus parsed on the target side, and use them to compute target-side tree features. We define additional string-to-tree features and, if a source-side dependency parser is available, tree-to-tree features to capture properties of how phrase dependencies interact with reordering.

To leverage standard phrase-based features alongside our novel features, we require a formalism that supports flexible feature combination and efficient decoding. Quasi-synchronous grammar (QG) provides this backbone (Smith and Eisner, 2006); we describe a coarse-to-fine approach for decoding within this framework, advancing substantially over earlier QG machine translation systems (Gimpel and Smith, 2009). The decoder involves generating a phrase lattice (Ueffing et al., 2002) in a coarse pass using a phrase-based model, followed by lattice dependency parsing of the phrase lattice. This approach allows us to feasibly explore the combined search space of segmentations, phrase alignments, and target phrase dependency trees.

Our experiments demonstrate an average improvement of +0.65 BLEU in Chinese-English translation across three test sets and an improvement of +0.75 BLEU in Urdu-English translation over a phrase-based baseline. We also describe experiments in which we replace supervised dependency parsers with *unsupervised* parsers, reporting promising results: using a supervised Chinese parser and a state-of-the-art unsupervised English parser provides our best results, giving an averaged gain of +0.79 BLEU over the baseline. We also discuss how our model improves translation quality and discuss future possibilities for combining approaches to ma-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 474–485,
Edinburgh, Scotland, UK, July 27–31, 2011.

chine translation using our framework.

2 Related Work

We previously applied quasi-synchronous grammar to machine translation (Gimpel and Smith, 2009), but that system performed translation fundamentally at the word level. Here we generalize that model to function on phrases, enabling a tighter coupling between the phrase segmentation and syntactic structures. We also present a decoder efficient enough to scale to large data sets and present performance improvements in large-scale experiments over a state-of-the-art phrase-based baseline.

Aside from QG, there have been many efforts to use dependency syntax in machine translation. Quirk et al. (2005) used a source-side dependency parser and projected automatic parses across word alignments in order to model dependency syntax on phrase pairs. Shen et al. (2008) presented an extension to Hiero (Chiang, 2005) in which rules have target-side dependency syntax and therefore enable the use of a dependency language model.

More recently, researchers have sought the benefits of dependency syntax while preserving the advantages of phrase-based models, such as efficiency and coverage. Galley and Manning (2009) loosened standard assumptions about dependency parsing so that the efficient left-to-right decoding procedure of phrase-based translation could be retained while a dependency language model is incorporated. Carreras and Collins (2009) presented a string-to-dependency system that permits non-projective dependency trees (thereby allowing a larger space of translations) and use a rule extraction procedure that includes rules for every phrase in the phrase table.

We take an additional step in this direction by working with dependency grammars on the phrases themselves, thereby bringing together the structural components of phrase-based and dependency-based MT in a single model. While others have worked on combining rules from multiple syntax-based systems (Liu et al., 2009) or using posteriors from multiple models to score translations (DeNero et al., 2010), we are not aware of any other work that seeks to directly integrate phrase-based and syntax-based machine translation at the modeling level.[1]

[1]Dymetman and Cancedda (2010) present a formal analysis of the problem of intersecting phrase-based and hierarchical translation models, but do not provide experimental results.

3 Model

Given a sentence $\boldsymbol{s}$ and its dependency tree $\tau_{\boldsymbol{s}}$, we formulate the translation problem as finding the target sentence $\boldsymbol{t}^*$, the segmentation $\boldsymbol{\gamma}^*$ of $\boldsymbol{s}$ into phrases, the segmentation $\boldsymbol{\phi}^*$ of $\boldsymbol{t}^*$ into phrases, the dependency tree $\tau^*_{\boldsymbol{\phi}}$ on the target phrases $\boldsymbol{\phi}^*$, and the one-to-one phrase alignment $\boldsymbol{a}^*$ such that

$$\langle \boldsymbol{t}^*, \boldsymbol{\gamma}^*, \boldsymbol{\phi}^*, \tau^*_{\boldsymbol{\phi}}, \boldsymbol{a}^* \rangle = \operatorname*{argmax}_{\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \rangle} p(\boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \,|\, \boldsymbol{s}, \tau_{\boldsymbol{s}})$$

We use a linear model (Och and Ney, 2002):

$$p(\boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}}) \propto \exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a})\}$$

where $\boldsymbol{g}$ is a vector of arbitrary feature functions on the full set of structures and $\boldsymbol{\theta}$ holds corresponding feature weights. Table 1 summarizes our notation.

In modeling $p(\boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}})$, we make use of **quasi-synchronous grammar** (QG; Smith and Eisner, 2006). Given a source sentence and its parse, a QG induces a probabilistic monolingual grammar over sentences "inspired" by the source sentence and tree. We denote this grammar by $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$; its (weighted) language is the set of translations of $\boldsymbol{s}$.

Quasi-synchronous grammar makes no restrictions on the form of the target monolingual grammar, though dependency grammars have been used in most previous applications of QG (Wang et al., 2007; Das and Smith, 2009; Smith and Eisner, 2009), including previous work in MT (Smith and Eisner, 2006; Gimpel and Smith, 2009). We previously presented a word-based machine translation model based on a quasi-synchronous dependency grammar. However, it is well-known in the MT community that translation quality is improved when larger units are modeled. Therefore, we use a dependency grammar in which the leaves are *phrases* rather than words.

We define a **phrase dependency grammar** as a model $p(\boldsymbol{\phi}, \tau_{\boldsymbol{\phi}} | \boldsymbol{t})$ over the joint space of segmentations of a sentence into phrases and dependency trees on the phrases.[2] Phrase dependency grammars

[2]We restrict our attention to projective trees in this paper, but the generalization to non-projective trees is easily made.

$\boldsymbol{s} = \langle s_1, \ldots, s_n \rangle$	source language sentence
$\boldsymbol{t} = \langle t_1, \ldots, t_m \rangle$	target language sentence, translation of $\boldsymbol{s}$
$\boldsymbol{\gamma} = \langle \gamma_1, \ldots, \gamma_{n'} \rangle$ $\forall i, \gamma_i = \langle s_j, \ldots, s_k \rangle$ s.t. $\gamma_1 \cdot \ldots \cdot \gamma_{n'} = \boldsymbol{s}$	segmentation of $\boldsymbol{s}$ into phrases
$\boldsymbol{\phi} = \langle \phi_1, \ldots, \phi_{m'} \rangle$ $\forall i, \phi_i = \langle t_j, \ldots, t_k \rangle$ s.t. $\phi_1 \cdot \ldots \cdot \phi_{m'} = \boldsymbol{t}$	segmentation of $\boldsymbol{t}$ into phrases
$\tau_{\boldsymbol{s}} : \{1, \ldots, n\} \rightarrow \{0, \ldots, n\}$	dependency tree on source words $\boldsymbol{s}$, where $\tau_{\boldsymbol{s}}(i)$ is the index of the parent of word s_i (0 is the root, \$)
$\tau_{\boldsymbol{\phi}} : \{1, \ldots, m'\} \rightarrow \{0, \ldots, m'\}$	dependency tree on target phrases $\boldsymbol{\phi}$, where $\tau_{\boldsymbol{\phi}}(i)$ is the index of the parent of phrase ϕ_i
$\boldsymbol{a} : \{1, \ldots, m'\} \rightarrow \{1, \ldots, n'\}$	one-to-one alignment from phrases in $\boldsymbol{\phi}$ to phrases in $\boldsymbol{\gamma}$
$\boldsymbol{\theta} = \langle \boldsymbol{\lambda}, \boldsymbol{\psi} \rangle$	parameters of the full model ($\boldsymbol{\lambda}$ = phrase-based, $\boldsymbol{\psi}$ = QPDG)

Table 1: Key notation.

have recently been used by Wu et al. (2009) for feature extraction for opinion mining. When used for translation modeling, they allow us to capture phenomena like local reordering and idiomatic translations within each phrase as well as long-distance relationships among the phrases in a sentence.

We then define a **quasi-synchronous phrase dependency grammar** (QPDG) as a conditional model $p(\boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}})$ that induces a probabilistic monolingual phrase dependency grammar over sentences inspired by the source sentence and (lexical) dependency tree. The source and target sentences are segmented into phrases and the phrases are aligned in a one-to-one alignment.

We note that we actually depart here slightly from the original definition of QG. The alignment variable in QG links target tree nodes to source tree nodes. However, we never commit to a source phrase dependency tree, instead using a source lexical dependency tree output by a dependency parser, so our alignment variable $\boldsymbol{a}$ is a function from target tree nodes (phrases in $\boldsymbol{\phi}$) to source phrases in $\boldsymbol{\gamma}$, which might not be source tree nodes. The features in our model may consider a large number of source phrase dependency trees as long as they are consistent with $\tau_{\boldsymbol{s}}$.

4 Features

Our model contains all of the standard phrase-based features found in systems like Moses (Koehn et al., 2007), including four phrase table probability features, a phrase penalty feature, an n-gram language model, a distortion cost, six lexicalized reordering features, and a word penalty feature.

We now describe in detail the additional features in our model that are used to score phrase dependency trees. We shall refer to these as QPDG features and will find it useful later to notationally distinguish their feature weights from those of the phrase-based model. We use $\boldsymbol{\lambda}$ for weights of the standard phrase-based model features and $\boldsymbol{\psi}$ for weights of the QPDG features. We include three categories of features, differentiated by what pieces of structure they consider.

\$ ← said :	\$ ← we should
\$ ← said that	\$ ← has been
\$ ← is a	- us → relations
\$ ← will be	\$ ← he said
\$ ← it is	cross - strait → relations
\$ ← this is	\$ ← pointed out that
\$ ← we must	, and → is
the → united states	the chinese → government
the → development of	\$ ← is the
the two → countries	\$ ← said ,
he → said :	one - china → principle
\$ ← he said :	sino - us → relations

Table 2: Most frequent phrase dependencies with at least 2 words in one of the phrases (dependencies in which one phrase is entirely punctuation are not shown). \$ indicates the root of the tree.

4.1 Target Tree Features

We first include features that only consider $\boldsymbol{t}$, $\boldsymbol{\phi}$, and $\tau_{\boldsymbol{\phi}}$. These features can be categorized as "syntactic language model" features (Shen et al., 2008; Galley and Manning, 2009), though unlike previous work our features model both the phrase segmentation and dependency structure. Typically, these sorts of features are probabilities estimated from a corpus parsed using a supervised parser. However, there do not currently exist treebanks with annotated phrase

, → made up	0.057
he → made up	0.021
supreme court → made up	0.014
court → made up	0.014
in september 2000 → made up	0.014
in september 2000 , → made up	0.014
made up ← of	0.065
made up ← .	0.029
made up ← ,	0.016
made up ← mind to	0.01

Table 3: Most probable child phrases for the parent phrase "made up" for each direction, sorted by the conditional probability of the child phrase given the parent phrase and direction.

dependency trees.

Our solution is to use a standard supervised dependency parser and extract phrase dependencies using bilingual information.[3] We begin by obtaining symmetrized word alignments and extracting phrase pairs using the standard heuristic from phrase-based MT (Koehn et al., 2003). Given the set of extracted phrase pairs for a sentence, denote by W the set of unique target-side phrases among them. We parse the target sentence with a dependency parser and, for each pair of phrases $u, v \in W$, we extract a phrase dependency (along with its direction) if u and v do not overlap and there is at least one lexical dependency between a word in u and a word in v. If there are lexical dependencies in both directions, we extract a phrase dependency only for the single longest one. Since we use a projective dependency parser, the longest lexical dependency between two phrases is guaranteed to be unique. Table 2 shows a listing of the most frequent phrase dependencies extracted (lexical dependencies are omitted).

We note that during training we never explicitly commit to any single phrase dependency tree for a target sentence. Rather, we extract phrase dependencies from all phrase dependency trees consistent with the word alignments and the lexical dependency tree. Thus we treat phrase dependency trees analogously to phrase segmentations in standard phrase extraction.

We perform this procedure on all sentence pairs in the parallel corpus. Given a set of extracted phrase dependencies of the form $\langle u, v, d \rangle$, where u is the head phrase, v is the child phrase, and $d \in \{left, right\}$ is the direction, we then estimate conditional probabilities $p(v|u, d)$ using relative frequency estimation. Table 3 shows the most probable child phrases for an example parent phrase. To combat data sparseness, we perform the same procedure with each word replaced by its word cluster ID obtained from Brown clustering (Brown et al., 1992).

We include a feature in the model for the sum of the scaled log-probabilities of each attachment:

$$\sum_{i=1}^{m'} \max\left(0, C + \log p(\phi_i | \phi_{\tau_{\boldsymbol{\phi}}(i)}, d(i)\right) \quad (1)$$

where $d(i) = I[\tau_{\boldsymbol{\phi}}(i) - i > 0]$ is the direction of the dependency arc.

Although we use log-probabilities in this feature function, we first add a constant C to each to ensure they are all positive.[4] The $\max$ expression protects unseen parent-child phrase dependencies from causing the score to be negative infinity. Our motivation is a desire for the features to be used to prefer one derivation over another but not to rule out a derivation completely if it merely happens to contain a dependency unobserved in the training data.

We also include lexical weighting features similar to those used in phrase-based MT (Koehn et al., 2003). Whenever we extract a phrase dependency, we extract the longest lexical dependency contained within it. For all ⟨parent, child, direction⟩ lexical dependency tuples $\langle x, y, d \rangle$, we estimate conditional probabilities $p_{lex}(y|x, d)$ from the parsed corpus using relative frequency estimation. Then, for a phrase dependency with longest lexical dependency $\langle x, y, d \rangle$, we add a feature for $p_{lex}(y|x, d)$ to the model, using a formula similar to Eq. 1. Different instances of a phrase dependency may have different lexical dependencies extracted with them. We add the lexical weight for the most frequent, breaking ties by choosing the lexical dependency that maximizes $p(y|x, d)$, as was also done by Koehn et al. (2003).

In all, we include 4 target tree features: one for phrase dependencies, one for lexical dependencies,

[3] For a monolingual task, Wu et al. (2009) used a shallow parser to convert lexical dependencies from a dependency parser into phrase dependencies.

[4] The reasoning here is that whenever we use a phrase dependency that we have observed in the training data, we want to boost the score of the translation. If we used log-probabilities, each observed dependency would incur a penalty.

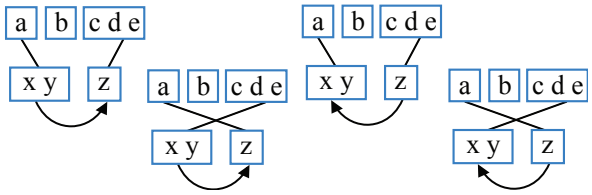


Figure 1: String-to-tree configurations; each is associated with a feature that counts its occurrences in a derivation.

Input: sentence $\boldsymbol{s}$, dependency parse $\tau_{\boldsymbol{s}}$, coarse parameters $\boldsymbol{\lambda}_M$, fine parameters $\langle \boldsymbol{\lambda}, \boldsymbol{\psi} \rangle$
Output: translation $\boldsymbol{t}$

$L_{\text{MERT}} \leftarrow \texttt{GenerateLattices}\,(\boldsymbol{s}, \boldsymbol{\lambda}_M)$;
$L_{\text{FB}} \leftarrow \texttt{FBPrune}\,(L_{\text{MERT}}, \boldsymbol{\lambda}_M)$;
$\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \rangle \leftarrow \text{QGDepParse}(L_{\text{FB}}, \langle \boldsymbol{\lambda}, \boldsymbol{\psi} \rangle)$;
return $\boldsymbol{t}$;

Algorithm 1: `CoarseToFineDecode`

and the same features computed from a transformed version of the corpus in which each word is replaced by its Brown cluster.

4.2 String-to-Tree Configurations

We consider features that count instances of reordering configurations involving phrase dependencies. In addition to the target-side structures, these features consider $\boldsymbol{\gamma}$ and $\boldsymbol{a}$, though not $\boldsymbol{s}$ or $\tau_{\boldsymbol{s}}$. For example, when building a parent-child phrase dependency with the child to the left, one feature value is incremented if their aligned source-side phrases are in the same order. This configuration is the leftmost in Fig. 1; we include features for the other three configurations there as well, for a total of 4 features in this category.

4.3 Tree-to-Tree Configurations

We include features that consider $\boldsymbol{s}$, $\boldsymbol{\gamma}$, and $\tau_{\boldsymbol{s}}$ in addition to $\boldsymbol{t}$, $\boldsymbol{\phi}$, and $\tau_{\boldsymbol{\phi}}$. We begin with features for each of the quasi-synchronous configurations from Smith and Eisner (2006), adapted to phrase dependency grammars. That is, for a parent-child pair $\langle \tau_{\boldsymbol{\phi}}(i), i \rangle$ in $\tau_{\boldsymbol{\phi}}$, we consider the relationship between $\boldsymbol{a}(\tau_{\boldsymbol{\phi}}(i))$ and $\boldsymbol{a}(i)$, the source-side phrases to which $\tau_{\boldsymbol{\phi}}(i)$ and i align. We use the following named configurations from Smith and Eisner: root-root, parent-child, child-parent, grandparent-grandchild, sibling, and c-command.[5] We define a feature to count instances of each of these configurations, including an additional feature for "other" configurations that do not fit into these categories.[6]

When using a QPDG, there are multiple ways to compute tree-to-tree configuration features, since we use a phrase dependency tree for the target side, a lexical dependency tree for the source side, and a phrase alignment. We use the following heuristic approach. Given a pair of source words, one with index j in source phrase $\boldsymbol{a}(\tau_{\boldsymbol{\phi}}(i))$ and the other with index k in source phrase $\boldsymbol{a}(i)$, we have a parent-child configuration if $\tau_{\boldsymbol{s}}(k) = j$; if $\tau_{\boldsymbol{s}}(j) = k$, a child-parent configuration is present. In order for the grandparent-grandchild configuration to be present, the intervening parent word must be outside both phrases. For sibling and other c-command configurations, the shared parent or ancestor must also be outside both phrases.

After obtaining a list of all configurations present for each pair of words $\langle j, k \rangle$, we fire the feature for the single configuration corresponding to the maximum distance $|j - k|$. If no configurations are present between any pair of words, the "other" feature fires. Therefore, only one configuration feature fires for each phrase dependency attachment.

Finally, we include features that consider the dependency path distance between phrases in the source-side dependency tree that are aligned to parent-child pairs in $\tau_{\boldsymbol{\phi}}$. We include a feature that sums, for each target phrase i, the inverse of the minimum undirected path length between each word in $\boldsymbol{a}(i)$ and each word in $\tau_{\boldsymbol{\phi}}(\boldsymbol{a}(i))$. The minimum undirected path length is defined as the number of dependency arcs that must be crossed to travel from one word to the other in $\tau_{\boldsymbol{s}}$. We use one feature for undirected path length and one other for directed path length. If there is no (un)directed path from a word in $\boldsymbol{a}(i)$ to a word in $\tau_{\boldsymbol{\phi}}(\boldsymbol{a}(i))$, we use ∞ as the minimum length.

There are 15 features in this category, for a total of 23 QPDG features.

[5] See Fig. 3 in Smith and Eisner (2006) for illustrations.

[6] We actually include two versions of each configuration feature other than "root-root": one for the source phrases being in the same order as the target phrases and one for them being swapped.

5 Decoding

For a QPDG model, decoding consists of finding the highest-scoring tuple $\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \tau_{\boldsymbol{\phi}}, \boldsymbol{a} \rangle$ for an input sentence $\boldsymbol{s}$ and its parse $\tau_{\boldsymbol{s}}$, i.e., finding the most probable derivation under the $\boldsymbol{s}/\tau_{\boldsymbol{s}}$-specific grammar $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$. We follow Gimpel and Smith (2009) in constructing a lattice to represent $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$ and using lattice parsing to search for the best derivation, but we construct the lattice differently and employ a coarse-to-fine strategy (Petrov, 2009) to speed up decoding.

It has become common in recent years for MT researchers to exploit efficient data structures for encoding concise representations of the pruned search space of the model, such as phrase lattices for phrase-based MT (Ueffing et al., 2002; Macherey et al., 2008; Tromble et al., 2008). Each edge in a phrase lattice corresponds to a phrase pair and each path through the lattice corresponds to a tuple $\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{a} \rangle$ for the input $\boldsymbol{s}$. Decoding for a phrase lattice consists of finding the highest-scoring path, which is done using dynamic programming. To also maximize over $\tau_{\boldsymbol{\phi}}$, we perform lattice dependency parsing, which allows us to search over the space of tuples $\langle \boldsymbol{t}, \boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{a}, \tau_{\boldsymbol{\phi}} \rangle$. Given the lattice and $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$, lattice parsing is a straightforward generalization of the standard arc-factored dynamic programming algorithm from Eisner (1996).

The lattice parsing algorithm requires $O(E^2V)$ time and $O(E^2 + VE)$ space, where E is the number of edges in the lattice and V is the number of nodes.[7] Typical phrase lattices might easily contain tens of thousands of nodes and edges, making exact search prohibitively expensive for all but the smallest lattices. So, we use approximate search based on coarse-to-fine decoding. We now discuss each step of this procedure; an outline is shown as Alg. 1.

Pass 1: Lattice Pruning After generating phrase lattices using a phrase-based MT system, we prune lattice edges using forward-backward pruning (Sixtus and Ortmanns, 1999), which has also been used in previous work using phrase lattices (Tromble et al., 2008). This pruning method computes the max-marginal for each lattice edge, which is the score of the best full path that uses that edge. Max-marginals offer the advantage that the best path in the lattice is preserved during pruning. For each lattice, we use a grid search to find the most liberal threshold that leaves fewer than 1000 edges in the resulting lattice. As complexity is quadratic in E, forcing E to be less than 1000 improves runtime substantially. After pruning, the lattices still contain more than 10^{16} paths on average and oracle BLEU scores are typically 12-15 points higher than the model-best paths.

Pass 2: Parent Ranking Given a pruned lattice, we then remove some candidate dependency arcs from consideration. It is common in dependency parsing to use a coarse model to rank the top k parents for each word, and to only consider these during parsing (Martins et al., 2009; Bergsma and Cherry, 2010). Unlike string parsing, our phrase lattices impose several types of constraints on allowable arcs. For example, each node in the phrase lattice is annotated with a coverage vector—a bit vector indicating which words in the source sentence have been translated—which implies a topological ordering of the nodes. To handle constraints like these, we first use the Floyd-Warshall algorithm (Floyd, 1962) to find the best score between every pair of nodes in the lattice. This algorithm also tells us whether each edge is reachable from each other edge, allowing us to immediately prune dependency arcs between edges that are unreachable from each other.

After eliminating impossible arcs, we turn to pruning away unlikely ones. In standard (string) dependency parsing, every word is assigned a parent. In lattice parsing, however, most lattice edges will not be assigned any parent. Certain lattice edges are much more likely to be contained within paths, so we allow some edges to have more candidate parent edges than others. We introduce hyperparameters α, β, and μ to denote, respectively, the minimum, maximum, and average number of parent edges to be considered for each lattice edge ($\alpha \leq \mu \leq \beta$). We rank the full set of E^2 arcs according to their scores (using the QPDG features and their weights $\boldsymbol{\psi}$) and choose the top μE of these arcs while ensuring that each edge has at least α and at most β potential parent edges.

This step reduces the time complexity from $O(E^2V)$ to $O(\mu EV)$, where $\mu < E$. In our experiments, we set $\mu = 300$, $\alpha = 100$, and $\beta = 400$.

[7] To prevent confusion, we use the term *edge* to refer to a phrase lattice edge and *arc* to refer to a parent-child dependency in the phrase dependency tree.

Input: tuning set $D = \langle S, T \rangle$, initial weights $\boldsymbol{\lambda}_0$ for coarse model, initial weights $\boldsymbol{\psi}_0$ for additional features in fine model
Output: coarse model learned weights: $\boldsymbol{\lambda}_M$, fine model learned weights: $\langle \boldsymbol{\lambda}^*, \boldsymbol{\psi}^* \rangle$

$\boldsymbol{\lambda}_M \leftarrow$ `MERT` (S, T, $\boldsymbol{\lambda}_0$, 100, MOSES);
$L_{\text{MERT}} \leftarrow$ `GenerateLattices` (S, $\boldsymbol{\lambda}_M$);
$L_{\text{FB}} \leftarrow$ `FBPrune` (L_{MERT}, $\boldsymbol{\lambda}_M$);
$\langle \boldsymbol{\lambda}^*, \boldsymbol{\psi}^* \rangle \leftarrow$
  `MERT` (L_{FB}, T, $\langle \boldsymbol{\lambda}_M, \boldsymbol{\psi}_0 \rangle$, 200, QGDEPPARSE);
return $\boldsymbol{\lambda}_M$, $\langle \boldsymbol{\lambda}^*, \boldsymbol{\psi}^* \rangle$;

Algorithm 2: `CoarseToFineTrain`

Pass 3: Lattice Dependency Parsing After completing the coarse passes, we parse using bottom-up dynamic programming based on the agenda algorithm (Nederhof, 2003; Eisner et al., 2005). We only consider arcs that survived the filtering in Pass 2. We weight agenda items by the sum of their scores and the Floyd-Warshall best path scores both from the start node of the lattice to the beginning of the item and the end of the item to any final node. This heuristic helps us to favor exploration of items that are highly likely under the phrase-based model.

If the score of the partial structure can only get worse when combining it with other structures (e.g., in a PCFG), then the first time that we pop an item of type GOAL from the agenda, we are guaranteed to have the best parse. However, in our model, some features are positive and others negative, making this property no longer hold; as a result, GOAL items may be popped out of order from the agenda. Therefore, we use an approximation, simply popping G GOAL items from the agenda and then stopping. The items are sorted by their scores and the best is returned by the decoder (or the k best in the case of MERT). In our experiments, we set $G = 4000$.

The combined strategy yields average decoding times in the range of 30 seconds per sentence, which is comparable to other syntax-based MT systems.

6 Training

For tuning the coarse and fine parameters, we use minimum error rate training (MERT; Och, 2003) in a procedure shown as Alg. 2. We first use MERT to train parameters for the coarse phrase-based model used to generate phrase lattices. Then, after generating the lattices, we prune them and run MERT a second time to tune parameters of the fine model, which includes all phrase-based and QPDG parameters. The arguments to `MERT` are a vector of source sentences (or lattices), a vector of target sentences, the initial parameter values, the size of the k-best list, and finally the decoder. We initialize $\boldsymbol{\lambda}$ to the default Moses feature weights and for $\boldsymbol{\psi}$ we initialize the two target phrase dependency weights to 0.004, the two lexical dependency weights to 0.001, and the weights for all configuration features to 0.0. Our training procedure requires two executions of MERT, and the second typically takes more iterations to converge (10 to 20 is typical) than the first due to the use of a larger feature set and increased possibility for search error due to the enlarged search space.

7 Experiments

For experimental evaluation, we consider Chinese-to-English (ZH-EN) and Urdu-to-English (UR-EN) translation and compare our system to Moses (Koehn et al., 2007). For ZH-EN, we used 303k sentence pairs from the FBIS corpus (LDC2003E14). We segmented the Chinese data using the Stanford Chinese segmenter in "CTB" mode (Chang et al., 2008), giving us 7.9M Chinese words and 9.4M English words. For UR-EN, we used parallel data from the NIST MT08 evaluation consisting of 1.2M Urdu words and 1.1M English words.

We trained a baseline Moses system using default settings and features. Word alignment was performed using GIZA++ (Och and Ney, 2003) in both directions and the `grow-diag-final-and` heuristic was used to symmetrize the alignments. We used a max phrase length of 7 when extracting phrases. Trigram language models were estimated using the SRI language modeling toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998). To estimate language models for each language pair, we used the English side of the parallel corpus concatenated with 200M words of randomly-selected sentences from the Gigaword v4 corpus (excluding the NY Times and LA Times).

We used this baseline Moses system to generate phrase lattices for our system, so our model includes all of the Moses features in addition to the

	MT03 (tune)	MT02	MT05	MT06	Average
Moses	33.84	33.35	31.81	28.82	31.33
QPDG (TT)	34.63 (+0.79)	34.10 (+0.75)	32.15 (+0.34)	29.33 (+0.51)	31.86 (+0.53)
QPDG (TT+S2T+T2T)	34.98 (+1.14)	34.26 (+0.91)	32.34 (+0.53)	29.35 (+0.53)	31.98 (+0.65)

Table 4: Chinese-English Results (% BLEU).

QPDG features described in §4. In our experiments, we compare our QPDG system (lattice parsing on each lattice) to the Moses baseline (finding the best path through each lattice). The conventional wisdom holds that hierarchical phrase-based translation (Chiang, 2005) performs better than phrase-based translation for language pairs that require large amounts of reordering, such as ZH-EN and UR-EN. However, researchers have shown that this performance gap diminishes when using a larger distortion limit (Zollmann et al., 2008) and may disappear entirely when using a lexicalized reordering model (Lopez, 2008; Galley and Manning, 2010). So, we increase the Moses distortion limit from 6 (the default) to 10 and use Moses' default lexicalized reordering model (Koehn et al., 2005).

We parsed the Chinese text using the Stanford parser (Levy and Manning, 2003) and the English text using TurboParser (Martins et al., 2009). We note that computing our features requires parsing the target (English) side of the parallel text, but not the source side. We only need to parse the source side of the tuning and test sets, and the only features that look at the source-side parse are those from §4.3.

To obtain Brown clusters for the target tree features in §4.1, we used code from Liang (2005).[8] We induced 100 clusters from the English side of the parallel corpus concatenated with 10M words of randomly-selected Gigaword sentences. Only words that appeared at least twice in this data were considered during clustering. An additional cluster was created for all other words; this allowed us to use phrase dependency cluster features even for out-of-vocabulary words. We used a max phrase length of 7 when extracting phrase dependencies to match the max phrase length used in phrase extraction. Approximately 87M unique phrase dependencies were extracted from the ZH-EN data and 7M from the UR-EN data.

We tuned the weights of our model using the procedure described in §6. For ZH-EN we used MT03 for tuning and MT02, MT05, and MT06 for testing. For UR-EN we used half of the documents (882 sentence pairs) from the MT08 test set for tuning ("Dev") and MT09 for testing. We evaluated translation output using case-insensitive IBM BLEU (Papineni et al., 2001).

	Dev (tune)	MT09
Moses	24.21	23.56
QPDG (TT+S2T)	24.94 (+0.73)	24.31 (+0.75)

Table 5: Urdu-English Results (% BLEU).

7.1 Results

Results for ZH-EN and UR-EN translation are shown in Tables 4 and 5. We show results when using only the target tree features from §4.1 (TT), as well as when adding the string-to-tree features from §4.2 (S2T) and the tree-to-tree features from §4.3 (T2T). We note that T2T features are unavailable for UR-EN because we do not have an Urdu parser. We find that we can achieve moderate but consistent improvements over the baseline Moses system, for an average increase of 0.65 BLEU points for ZH-EN and 0.75 for UR-EN.

Fig. 2 shows an example sentence from the MT05 test set along with its translation output and derivations produced by Moses and our QPDG system with the full feature set. This example shows the kind of improvements that our system makes. In Chinese, modifiers such as prepositional phrases and clauses are generally placed in front of the words they modify, frequently the opposite of English. In addition, Chinese occasionally uses postpositions where English uses prepositions. The Chinese sentence in Fig. 2 exhibits both of these, as the prepositional phrase "after the Palestinian election" appears before the verb "strengthen" in the Chinese sentence and "after" appears as a postposition. Moses (Fig. 2(a)) does not properly reorder the prepositional phrase, while our system (Fig. 2(b)) properly handles both reorderings.[9] We shall discuss these

[8] http://www.cs.berkeley.edu/~pliang/software

[9] Our system's derivation is not perfect, in that "in" is incor-

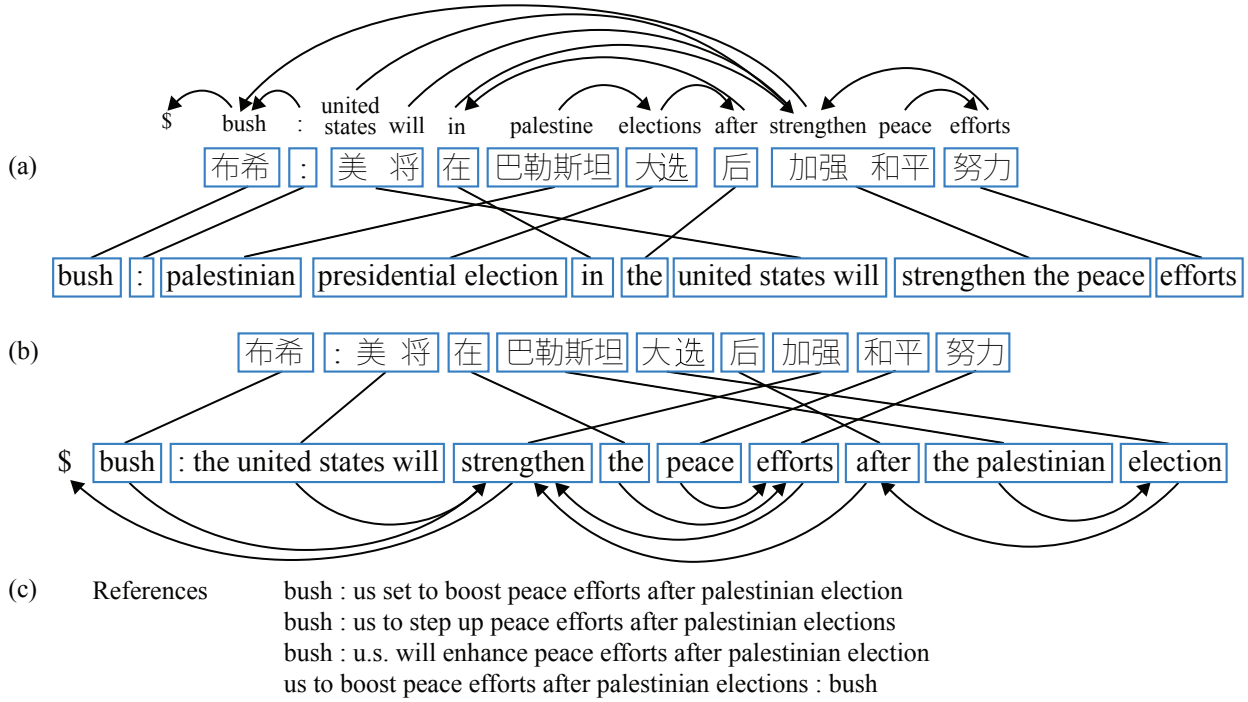


Figure 2: (a) Moses translation output along with γ, ϕ, and $\boldsymbol{a}$. An English gloss is shown above the Chinese sentence and above the gloss is shown the dependency parse from the Stanford parser. (b) QPDG system output with additional structure τ_{ϕ}. (c) reference translations.

types of improvements further in §8.

7.2 Unsupervised Parsing

Our results thus far use supervised parsers for both Chinese and English, but parsers are only available for a small fraction of the languages we would like to translate. Fortunately, unsupervised dependency grammar induction has improved substantially in recent years due to a flurry of recent research. While attachment accuracies on standard treebank test sets are still relatively low, it may be the case that even though unsupervised parsers do not match treebank annotations very well, they may perform well when used for extrinsic applications. We believe that syntax-based MT offers a compelling platform for development and extrinsic evaluation of unsupervised parsers.

In this paper, we use the standard dependency model with valence (DMV; Klein and Manning, 2004). When training is initialized using the output of a simpler, concave dependency model, the DMV can approach state-of-the-art unsupervised accuracy (Gimpel and Smith, 2011). For English, the resulting parser achieves 53.1% attachment accuracy on Section 23 of the Penn Treebank (Marcus et al., 1993), which approaches the 55.7% accuracy of a recent state-of-the-art unsupervised model (Blunsom and Cohn, 2010). The Chinese parser, initialized and trained the same way, achieves 44.4%, which is the highest reported accuracy on the Chinese Treebank (Xue et al., 2004) test set.

Most unsupervised grammar induction models assume gold standard POS tags and sentences stripped of punctuation. We use the Stanford tagger (Toutanova et al., 2003) to obtain tags for both English and Chinese, parse the sentences without punctuation using the DMV, and then attach punctuation tokens to the root word of the tree in a post-processing step. For English, the predicted parents agreed with those of TurboParser for 48.7% of the tokens in the corpus.

We considered all four scenarios: supervised and unsupervised English parsing paired with supervised and unsupervised Chinese parsing. Table 6 shows

rectly translated and reordered, but the system was nonetheless able to use it to improve the fluency of the output.

		EN	
		unsupervised	supervised
ZH	unsupervised	31.18 (33.76)	31.86 (34.78)
	supervised	**32.12** (34.74)	31.98 (34.98)
Moses		31.33 (33.84)	

Table 6: Results when using unsupervised dependency parsers. Cells contain averaged % BLEU on the three test sets and % BLEU on tuning data (MT03) in parentheses.

Feature	Initial	Learned
Left child, same order	9.0	8.9
Left child, swap phrases	1.1	0.0
Right child, same order	7.3	7.3
Right child, swap phrases	1.6	2.3
Root-root	0.4	0.8
Parent-child	4.2	6.1
Child-parent	1.2	0.4
Grandparent-grandchild	1.0	0.2
Sibling	2.4	1.9
C-command	6.1	6.7
Other	1.5	0.9

Table 7: Average feature values across best translations of sentences in the MT03 tuning set, both before MERT (column 2) and after (column 3). "Same" versions of tree-to-tree configuration features are shown; the rarer "swap" features showed a similar trend.

BLEU scores averaged over the three test sets with tuning data BLEU in parentheses. Surprisingly, we achieve our best results when using the unsupervised English parser in place of the supervised one (+0.79 over Moses), while keeping the Chinese parser supervised. Competitive performance is also found by using the unsupervised Chinese parser and supervised English parser (+0.53 over Moses).

However, when using unsupervised parsers for both languages, performance was below that of Moses. During tuning for this configuration, we found that MERT struggled to find good parameter estimates, typically converging to suboptimal solutions after a small number of iterations. We believe this is due to the large number of features (37), the noise in the parse trees, and known instabilities of MERT. In future work we plan to experiment with training algorithms that are more stable and that can handle larger numbers of features.

8 Analysis

To understand what our model learns during MER training, we computed the feature vectors of the best derivation for each sentence in the tuning data at both the start and end of tuning. Table 7 shows these feature values averaged across all tuning sentences. The first four features are the configurations from Fig. 1, in order from left to right. From these rows, we can observe that the model learns to encourage swapping when generating right children and penalize swapping for left children. In addition to objects, right children in English are often prepositional phrases, relative clauses, or other modifiers; as we noted above, Chinese generally places these modifiers before their heads, requiring reordering during translation. Here the model appears to be learning this reordering behavior.

From the second set of features, we see that the model learns to favor producing dependency trees that are mostly isomorphic to the source tree, by favoring root-root and parent-child configurations at the expense of most others.

9 Discussion

In looking at BLEU score differences between the two systems, the unigram precisions were typically equal or only slightly different, while precisions for higher-order n-grams contained the bulk of the improvement. This suggests that our system is not finding substantially better translations for individual words in the input, but rather is focused on reordering the existing translations. This is not surprising given our choice of features, which focus on syntactic language modeling and syntax-based reordering. The obvious next step for our framework is to include bilingual rules that include source syntax (Quirk et al., 2005), target syntax (Shen et al., 2008), and syntax on both sides. Our framework allows integrating together all of these and other types of structures, with the ultimate goal of combining the strengths of multiple approaches to translation in a single model.

Acknowledgments

We thank Chris Dyer and the anonymous reviewers for helpful comments that improved this paper. This research was supported in part by the NSF through grant IIS-0844507, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533, and Sandia National Laboratories (fellowship to K. Gimpel).

References

M. Auli, A. Lopez, H. Hoang, and P. Koehn. 2009. A systematic analysis of translation model search spaces. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

S. Bergsma and C. Cherry. 2010. Fast and accurate arc filtering for dependency parsing. In *Proc. of COLING*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proc. of EMNLP*.

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18.

X. Carreras and M. Collins. 2009. Non-projective parsing for statistical machine translation. In *Proc. of EMNLP*.

P. Chang, M. Galley, and C. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proc. of the Third Workshop on Statistical Machine Translation*.

S. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report 10-98, Harvard University.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL*.

D. Das and N. A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

S. DeNeefe, K. Knight, W. Wang, and D. Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. of EMNLP-CoNLL*.

J. DeNero, S. Kumar, C. Chelba, and F. J. Och. 2010. Model combination for machine translation. In *Proc. of NAACL*.

M. Dymetman and N. Cancedda. 2010. Intersecting hierarchical and phrase-based models of translation. formal aspects and algorithms. In *Proc. of SSST-4*.

J. Eisner, E. Goldlust, and N. A. Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

R. W. Floyd. 1962. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6).

M. Galley and C. D. Manning. 2009. Quadratic-time dependency parsing for machine translation. In *Proc. of ACL-IJCNLP*.

M. Galley and C. D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Proc. of NAACL*.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of COLING-ACL*.

K. Gimpel and N. A. Smith. 2009. Feature-rich translation by quasi-synchronous lattice parsing. In *Proc. of EMNLP*.

K. Gimpel and N. A. Smith. 2011. Concavity and initialization for unsupervised dependency grammar induction. Technical report, Carnegie Mellon University.

L. Huang, K. Knight, and A. Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*.

P. Koehn, A. Axelrod, A. Birch Mayne, C. Callison-Burch, M. Osborne, and D. Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *Proc. of IWSLT*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL (demo session)*.

R. Levy and C. D. Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proc. of ACL*.

P. Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Y. Liu, H. Mi, Y. Feng, and Q. Liu. 2009. Joint decoding with multiple translation models. In *Proc. of ACL-IJCNLP*.

A. Lopez. 2008. Tera-scale translation models via pattern matching. In *Proc. of COLING*.

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *EMNLP*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

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

M.-J. Nederhof. 2003. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29(1).

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*.

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

F. J. Och. 2003. Minimum error rate training for statistical machine translation. In *Proc. of ACL*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

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

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*.

L. Shen, J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL*.

A. Sixtus and S. Ortmanns. 1999. High quality word graphs using forward-backward pruning. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*.

D. A. Smith and J. Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of HLT-NAACL Workshop on Statistical Machine Translation*.

D. A. Smith and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous features. In *Proc. of EMNLP*.

A. Stolcke. 2002. SRILM—an extensible language modeling toolkit. In *Proc. of ICSLP*.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*.

R. Tromble, S. Kumar, F. Och, and W. Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *EMNLP*.

N. Ueffing, F. J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proc. of EMNLP*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.

Y. Wu, Q. Zhang, X. Huang, and L. Wu. 2009. Phrase dependency parsing for opinion mining. In *Proc. of EMNLP*.

N. Xue, F. Xia, F.-D. Chiou, and M. Palmer. 2004. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

A. Zollmann and A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of NAACL 2006 Workshop on Statistical Machine Translation*.

A. Zollmann, A. Venugopal, F. J. Och, and J. Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proc. of COLING*.

A Word Reordering Model for Improved Machine Translation

Karthik Visweswariah
IBM Research India
Bangalore, India
v-karthik@in.ibm.com

Rajakrishnan Rajkumar
Dept. of Linguistics
Ohio State University
raja@ling.osu.edu

Ankur Gandhe
IBM Research India
Bangalore, India
ankugand@in.ibm.com

Ananthakrishnan Ramanathan
IBM Research India
Bangalore, India
aramana2@in.ibm.com

Jiri Navratil
IBM T.J. Watson Research Center
Yorktown Heights, New York
jiri@us.ibm.com

Abstract

Preordering of source side sentences has proved to be useful in improving statistical machine translation. Most work has used a parser in the source language along with rules to map the source language word order into the target language word order. The requirement to have a source language parser is a major drawback, which we seek to overcome in this paper. Instead of using a parser and then using rules to order the source side sentence we learn a model that can directly reorder source side sentences to match target word order using a small parallel corpus with high-quality word alignments. Our model learns pairwise costs of a word immediately preceding another word. We use the Lin-Kernighan heuristic to find the best source reordering efficiently during training and testing and show that it suffices to provide good quality reordering.

We show gains in translation performance based on our reordering model for translating from Hindi to English, Urdu to English (with a public dataset), and English to Hindi. For English to Hindi we show that our technique achieves better performance than a method that uses rules applied to the source side English parse.

1 Introduction

Languages differ in the way they order words to produce sentences representing the same meaning. Machine translation systems need to reorder words in the source sentence to produce fluent output in the target language that preserves the meaning of the source sentence.

Current phrase based machine translation systems can capture short range reorderings via the phrase table. Even the capturing of these local reordering phenomena is constrained by the amount of training data available. For example, if adjectives precede nouns in the source language and follow nouns in the target language we still need to see a particular adjective noun pair in the parallel corpus to handle the reordering via the phrase table. Phrase based systems also rely on the target side language model to produce the right target side order. This is known to be inadequate (Al-Onaizan and Papineni, 2006), and this inadequacy has spurred various attempts to overcome the problem of handling differing word order in languages.

One approach is through distortion models, that try to model which reorderings are more likely than others. The simplest models just penalize long jumps in the source sentence when producing the target sentence. These models have also been generalized (Al-Onaizan and Papineni, 2006; Tillman, 2004) to allow for lexical dependencies on the source. While these models are simple, and can be integrated with the decoder they are insufficient to capture long-range reordering phenomena especially for language pairs that differ significantly.

The weakness of these simple distortion models has been overcome using syntax of either the source or target sentence (Yamada and Knight, 2002; Galley et al., 2006; Liu et al., 2006; Zollmann and Venugopal, 2006). While these methods have shown to be useful in improving machine translation perfor-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 486–496,
Edinburgh, Scotland, UK, July 27–31, 2011.

mance they generally involve joint parsing of the source and target language which is significantly more computationally expensive when compared to phrase based translation systems. Another approach that overcomes this weakness, is to to reorder the source sentence based on rules applied on the source parse (either hand written or learned from data) both when training and testing (Collins et al., 2005; Genzel, 2010; Visweswariah et al., 2010).

In this paper we propose a novel method for dealing with the word order problem that is efficient and does not rely on a source or target side parse being available. We cast the word ordering problem as a Traveling Salesman Problem (TSP) based on previous work on word-based and phrased-based statistical machine translation (Tillmann and Ney, 2003; Zaslavskiy et al., 2009). Words are the cities in the TSP and the objective is to learn the distance between words so that the shortest tour corresponds to the ordering of the words in the source sentence in the target language. We show that the TSP distances for reordering can be learned from a small amount of high-quality word alignment data by means of pairwise word comparisons and an informative feature set involving words and part-of-speech (POS) tags adapted and extended from prior work on dependency parsing (McDonald et al., 2005b). Obtaining high-quality word alignments that we require for training is fairly easy compared with obtaining a treebank required to obtain parses for use in syntax based methods.

We show experimentally that our reordering model, even when used to reorder sentences for training and testing (rather than being used as an additional score in the decoder) improves machine translation performance for: Hindi $\rightarrow$ English, English $\rightarrow$ Hindi, and Urdu $\rightarrow$ English. Although Urdu is similar to Hindi from the point of reordering phenomena we include it in our experiments since there are publicly available datasets for Urdu-English. For English $\rightarrow$ Hindi we obtained better machine translation performance with our reordering model as compared to a method that uses reordering rules applied to the source side parse.

The rest of the paper is organized as follows. Section 2 reviews related work and places our work in context. Section 3 outlines reordering issues due to syntactic differences between Hindi and English. Section 4 presents our reordering model, Section 5 presents experimental results and Section 6 presents our conclusions and possible future work.

2 Related work

There have been several studies demonstrating improved machine translation performance by reordering source side sentences based on rules applied to the source side parse during training and decoding. Much of this work has used hand written rules and several language pairs have been studied e.g German to English (Collins et al., 2005), Chinese to English (Wang et al., 2007), English to Hindi (Ramanathan et al., 2009), English to Arabic (Badr et al., 2009) and Japanese to English (Lee et al., 2010). There have also been some studies where the rules are learned from the data (Genzel, 2010; Visweswariah et al., 2010; Xia and McCord, 2004). In addition there has been work (Yamada and Knight, 2002; Zollmann and Venugopal, 2006; Galley et al., 2006; Liu et al., 2006) which uses source and/or target side syntax in a Context Free Grammar framework which results in machine translation decoding being considered as a parsing problem. In this paper we propose a model that does not require either source or target side syntax while also preserving the efficiency of reordering techniques based on rules applied to the source side parse.

In work that is closely related to ours, (Tromble and Eisner, 2009) formulated word reordering as a Linear Ordering Problem (LOP), an NP-hard permutation problem. They learned LOP model weights capable of assigning a score to every possible permutation of the source language sentence from an aligned corpus by using a averaged perceptron learning model. The key difference between our model and the model in (Tromble and Eisner, 2009) is that while they learn costs of a word w_i appearing *anywhere* before w_j, we learn costs of w_i *immediately preceding* w_j. This results in more compact models and (as we show in Section 5) better models.

Our model results in us having to solve a TSP instance. The relation between the TSP and machine translation decoding has been explored before. (Knight, 1999) showed that TSP is a sub-class of MT decoding and thus established that the latter is NP-hard. (Zaslavskiy et al., 2009) casts phrase-based

decoding as a TSP and they show favorable speed performance trade-offs compared with *Moses*, an existing state-of-the-art decoder. In (Tillmann and Ney, 2003), a beam-search algorithm used for TSP is adapted to work with an IBM-4 word-based model and phrase-based model respectively. As opposed to calculating TSP distances from existing machine translation components (viz. the translation, distortion and language model probabilities) we *learn model weights* to reorder source sentences to match target word order using an informative feature set adapted from graph-based dependency parsing (McDonald et al., 2005a).

3 Hindi-English reordering issues

This section provides a brief survey of constructions that the two languages in question differ as well as have in common. (Ramanathan et al., 2009) notes the following divergences:

- English follows SVO order while Hindi follows SOV order
- English uses prepositions while Hindi uses post-positions
- Hindi allows greater word order freedom
- Hindi has a relatively richer case-marking system

In addition to these differences, (Visweswariah et al., 2010) mention the similarity in word order in the case of adjective noun sequences (*some books* vs. *kuch kitab*).

4 Reordering model

Consider a source sentence $\mathbf{w}$ consisting of a sequence of n words w_1, w_2, ... w_n that we would like to reorder into the target language order. Given a permutation π of the indices $1..n$, let the candidate reordering be w_{π_1}, w_{π_2}, ..., w_{π_n}. Thus, π_i denotes the index of the word in the source sentence that maps to position i in the candidate reordering. Clearly there are $n!$ such permutations. Our reordering model assigns costs to candidate permutations as:

$$C(\pi|\mathbf{w}) = \sum_i c(\pi_{i-1}, \pi_i).$$

The cost $c(m, n)$ can be thought of as the cost of the word at index m immediately preceding the word with index n in the candidate reordering. In this paper, we parametrize the costs as:

$$c(m, n) = \theta^T \Phi(\mathbf{w}, m, n),$$

where θ is a learned vector of weights and Φ is a vector of feature functions.

Given a source sentence $\mathbf{w}$ we reorder it according to the permutation π that minimizes the cost $C(\pi|\mathbf{w})$. Thus, we would like our cost function $C(\pi|\mathbf{w})$ to be such that the correct reordering π^* has the lowest cost of all possible reorderings π. In Section 4.1 we describe the features Φ that we use, and in Section 4.2 we describe how we train the weights θ to obtain a good reordering model.

Given our model structure, the minimization problem that we need to solve is identical to solving a Asymmetric Traveling Salesman Problem (ATSP) with each word corresponding to a city, and the costs $c(m, n)$ representing the pairwise distances between the cities. Consider the following example:

English input: John eats apples

Hindi: John seba(apples) khaataa hai(eats)

Desired reordered English: John apples eats

The ATSP that we need to solve is represented pictorially in Figure 1 with sample costs. Note that we have one extra node numbered 0. We start and end the tour at node 0, and this determines the first word in the reordered sentence. In this example the minimum cost tour is:

Start $\rightarrow$ *John* $\rightarrow$ *apple* $\rightarrow$ *eats*

recovering the right reordering for translation into Hindi.

Solving the ATSP (which is a well known NP hard problem) efficiently is crucial for the efficiency of our reordering model. To solve the ATSP, we first convert the ATSP to a symmetric TSP and then use the Lin-Kernighan heuristic as implemented in *Concorde*, a state-of-the-art TSP solver (Applegate et al., 2005). We also experimented with using the exact TSP solver in *Concorde* but since it was slower and did not improve performance we preferred using the Lin-Kernighan heuristic. To convert the ATSP to a symmetric TSP we double the size of the original problem creating a node $N^{'}$ for every node N in the original graph. Following (Hornik and

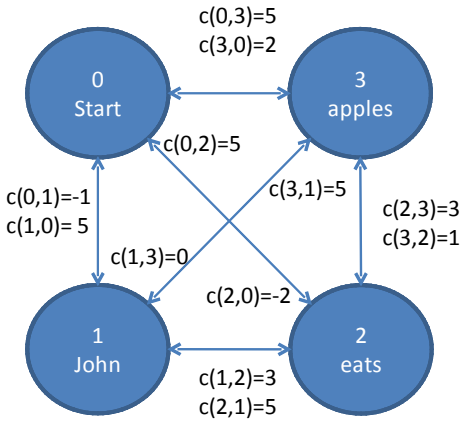


Figure 1: Example of an ATSP for reordering the sentence: *John eats apples*.

w_i	p_i	w_j	p_j
×	×	×	×
×			
	×		
		×	
			×
×	×		
×		×	
×			×
	×	×	
	×		×
		×	×
×	×		×

Table 1: Bigram feature templates used to calculate the cost that word at position i immediately precedes word at position j in the target word order. w_i (p_i) denotes the word (POS tag) at position i in the source sentence. Each of the templates is also conjoined with i-j the signed distance between the two words in the source sentence.

Hahsler, 2009), we then set new costs as follows: $c^{'}(A, B) = \infty$, $c^{'}(A, B^{'}) = c^{'}(B^{'}, A) = c(A, B)$ and $C(A, A^{'}) = -\infty$. Even with this doubling of the number of nodes, we observed that solving the TSPs with the Lin-Kernighan heuristic is very fast, taking roughly 10 milliseconds per sentence on average. Overall, this means that our reordering model is as fast as parsing and hence our model is comparable in performance to techniques based on applying rules to the parse tree.

4.1 Features

Since we would like to model reordering phenomena which are largely related to analyzing the syntax of the source sentence, we chose to use features based on those that have in the past been used for parsing (McDonald et al., 2005a). A subset of the features we use was also used for reordering in (Tromble and Eisner, 2009).

To be able to generalize from relatively small amounts of data, we use features that in addition to depending on the words in the input sentence $\mathbf{w}$ depend on the part-of-speech (POS) tags of the words in the input sentence. All features $\Phi(\mathbf{w}, i, j)$ we use are binary features, that fire based on the identities of the words and POS tags at or surrounding positions i and j in the source sentence. The first set of feature templates we use are given in Table 1. These features depend only on the identities of the word and POS tag of the two positions i and j and we call these Bigram features.

The second set of feature templates we use are given in Table 2. These features, in addition to examining positions i and j examine the surrounding positions. We instantiate these feature templates separately for the POS tag sequence and for the word sequence. We call these two feature sets ContextPOS and ContextWord respectively. When instantiated with POS tags, the first row of Table 2 looks at all POS tags between positions i and j. (Tromble and Eisner, 2009) use Bigram and ContextPOS features, while we extend their feature set with the use of ContextWord features. Since Hindi is verb final, in Hindi sentences with multiple verb groups it is rare for words with a verb in between to be placed together in the reordering to match English. Looking at the POS tags of words between positions i and j allows us to penalize such reorderings.

Each of the templates described in Table 1 and Table 2 is also conjoined with i-j the signed distance between the two words in the source sentence. The values of i-j between 5 and 10, and greater than 10 are quantized (negative values are similarly quantized).

In Section 5.2 we report on experiments showing the relative performance of these different feature

o_{i-1}	o_i	o_{i+1}	o_b	o_{j-1}	o_j	o_{j+1}
	×		×		×	
×	×				×	×
	×				×	×
×					×	×
×	×					×
×	×				×	
	×	×		×	×	
		×		×	×	
	×			×	×	
	×				×	
	×	×		×		
×	×			×	×	
×				×	×	
×	×			×		
	×	×			×	×
		×			×	×
	×	×				×

Table 2: Context feature templates used to calculate the cost that word at position i immediately precedes word at position j in the target word order. o_i denotes the observation at position i in the source sentence and o_b denotes an observation at a position between i and j (i.e $i+1 \leq b \leq j-1$). Each of the templates is instantiated with the observation sequence $\mathbf{o}$ taken to be the word sequence $\mathbf{w}$ and the POS tag sequence $\mathbf{p}$. Each of the templates is also conjoined with i-j the signed distance between the two positions in the source sentence.

types for the task of reordering Hindi sentences to be in English word order.

4.2 Training

To train the weights θ in our model, we need a collection of sentences, where we have the desired reference reordering $\pi^*(\mathbf{x})$ for each input sentence $\mathbf{x}$. To obtain these reference reorderings we use word aligned source-target sentence pairs. The quality and consistency of these reference reorderings will depend on the quality of the word alignments that we use. Given word aligned source and target sentences, we drop the source words that are not aligned. Let m_i be the mean of the target word positions that the source word at index i is aligned to. We then sort the source indices in increasing order of m_i. If $m_i = m_j$ (for example, because w_i and w_j are aligned to the same set of words) we keep them in the same order that they occurred in the source sentence. Obtaining the target ordering in this manner, is certainly not the only possible way and we would like to explore better treatment of this in future work.

We used the single best Margin Infused Relaxed Algorithm (MIRA) ((McDonald et al., 2005b), (Crammer and Singer, 2003)) with the online updates to our parameters being given by

$$\begin{aligned} \theta_{i+1} = & \arg\min_{\theta} ||\theta - \theta_i|| \\ & s.t. \quad C(\pi^*|\mathbf{w}) < C(\hat{\pi}|\mathbf{w}) - L(\pi^*, \hat{\pi}). \end{aligned}$$

In the equation above,

$$\hat{\pi} = \arg\min_{\pi} C(\pi|\mathbf{x})$$

is the best reordering based on the current parameter value and L is a loss function. We take the loss of a reordering to be the number of words for which the preceding word is wrong relative to the reference target order.

We also experimented with the averaged perceptron algorithm (Collins, 2002), but found single best MIRA to work slightly better and hence used MIRA for all our experiments.

5 Experiments

In this section we report on experiments to evaluate our reordering model. The first method we use for evaluation (monolingual BLEU) is by generating the desired reordering of the source sentence (as described in Section 4.2) and compare the reordered output to this desired reordered sentence using the BLEU metric. In addition, to these monolingual BLEU results, we also evaluate (in Section 5.5) the reordering by its effect on eventual machine translation performance.

We note that our reordering techniques uses POS information for the input sentence. The POS taggers used in this paper are Maximum Entropy Markov models trained using manually annotated POS corpora. For Hindi, we used roughly fifty thousand words with twenty six tags from the corpus described in (Dalal et al., 2007). For Urdu we used roughly fifty thousand words and forty six tags from the CRULP corpus (Hussain, 2008) and for English we used the Wall Street Journal section of the Penn Treebank.

5.1 Reordering model training data and alignment quality

To train our reordering models we need training data where we have the input source language sentence and the desired reordering in the target language. As described in Section 4.2 we derive the reference reordered sentence using word alignments. Table 3 presents our monolingual BLEU results for Hindi to English reordering as the source of the word alignments is varied. All results in Table 3 are with Bigram and ContextPOS features. We have word alignments from three sources: A small set of hand aligned sentences, HMM alignments (Vogel et al., 1996) and alignments obtained using a supervised Maximum Entropy aligner (Ittycheriah and Roukos, 2005) trained on the hand alignments. The F-measure for the HMM alignments were 65% and 78% for the Maximum Entropy model alignments. We see that the quality of the alignments is an important determiner of reordering performance. Row 1 shows the BLEU for unreordered (baseline) Hindi compared with the Hindi sentences reordered in English Order. Using just HMM alignments to train our model we do worse than unreordered Hindi. Although using the Maximum Entropy alignments is better than using HMM alignments, we do not improve upon a small number of hand alignments by using all the Maximum Entropy alignments.

To improve upon the model trained with only hand alignments we selected a small number of snippets of sentences from our Maximum Entropy alignments. The goal was to pick parts of sentences where the alignment is reliable enough to use for training. The heuristic we used in the selection of snippets was to pick maximal snippets of at least 7 consecutive Hindi words with all Hindi words aligned to a consecutive span of English words, with no unaligned English words in the span and no English words aligned to Hindi words outside the span. Adding snippets selected with this heuristic improves the reordering performance of our model as seen in the last row of Table 3.

5.2 Feature set comparison

In this section we report on experiments to determine the performance of the different classes of features (Bigram, ContextPos and ContextWord) discussed in Section 4.1. Table 4 shows monolingual BLEU results for training with different features sets for Hindi to English reordering. In all cases, we use a set of 6000 sentence pairs which were hand aligned to generate the training data. It is clear that all three sets of features contribute to performance of the reordering model, however the number of ContextWord features is larger than the number of Bigram and ContextPOS features put together, and it may be desirable to select from this set of features especially when training on large amounts of data.

HMM	MaxEnt	Hand	BLEU
-	-	-	35.9
220K	-	-	35.4
-	220K	-	47.0
-	220K	6K	48.4
-	-	6K	49.0
-	Good 17K	6K	**51.3**

Table 3: Monolingual BLEU scores for Hindi to English reordering using models trained on different alignment types and tested on a development set of 280 Hindi sentences (5590 tokens).

Feature template			
Bigram	ContextPOS	ContextWord	BLEU
-	-	-	35.9
×	-	-	43.8
×	×	-	49.0
×	×	×	**51.3**

Table 4: Monolingual BLEU scores for Hindi to English reordering using models trained with different feature sets and tested on a development set of 280 Hindi sentences (5590 tokens).

5.3 Monolingual reordering comparisons

Table 5 compares our reordering model with a reimplementation of the reordering model proposed in (Tromble and Eisner, 2009). Both the models use exactly the same features (bigram features and ContextPOS features) and are trained on the same data. To generate our training data, for Hindi to English and English to Hindi we use a set of 6000 hand aligned sentences, for Urdu to English we use a set of 8500 hand aligned sentences and for English to French we use a set of 10000 hand aligned sentences (a subset of Europarl and Hansards corpus). Our

Language pair		Monolingual BLEU		
Source	Target	Unreordered	LOP	TSP
Hindi	English	35.9	36.6	**49.0**
English	Hindi	34.4	48.4	**56.7**
Urdu	English	35.6	39.5	**49.9**
English	French	64.4	78.2	**81.2**

Table 5: Monolingual BLEU scores comparing the original source order with desired target reorder without reordering, and reordering using our model (TSP) and the model proposed in (Tromble and Eisner, 2009) (LOP).

test data consisted of 280 sentences for Hindi to English and 400 sentences for all other language pairs generated from hand aligned sentences. We include English-French here to compare on a fairly similar language pair with local reordering phenomena (the main difference being that in French adjectives generally follow nouns). We note that our model outperforms the model proposed in (Tromble and Eisner, 2009) in all cases.

5.4 Analysis of reordering performance

To get a feel for the qualitative performance of our reordering algorithm and the kind of phenomena it is able to capture, we analyze the reordering performance in terms of (i) whether the clause restructuring is done correctly – these can be thought of as medium-to-long range reorderings, (ii) whether clause boundaries are respected, and (iii) whether local (short range) reordering is performed correctly. The following analysis is for Hindi to English reordering with the best model (this is also the model used for Machine Translation experiments reported on in Section 5.5).

- **Clause structure**: As discussed in Section 3, the canonical clause order in Hindi is SOV, while in English it is SVO. However, variations on this structure are possible and quite frequent (e.g., clauses with two objects). To evaluate clause restructuring, we compared sequences of subjects, objects and verbs in the output and reference reorderings.

  We had a set of 70 sentences annotated with subject, direct object, indirect object and verb information – these annotations were made on the head word of each phrase, and the comparisons were on sequences of these words alone and not the entire constituent phrase. 52 sentences were reordered by the model to match the order of the corresponding reference. Eight sentences were ordered correctly but differently from the reference, because the reference was expressed in non-canonical fashion (e.g., in the passive) – note that these cases negatively impact the monolingual BLEU score. The following example shows a sentence being reordered correctly, where, however, the reference is expressed differently (note the position of the subject "policy" (*niiti*) in the reference and the reordered output) [1]:

  Input: aba$_1$ (now) taka$_2$ (till) aisii$_3$ (this) *niiti*$_4$ (policy) kabhii$_5$ (ever) nahii$_6$ (not) rahii$_7$ (has) hai$_8$ (been)

  Reordered: taka$_2$ (till) aba$_1$ (now) aisii$_3$ (this) *niiti*$_4$ (policy) hai$_8$ (been) kabhii$_5$ (ever) nahii$_6$ (not) rahii$_7$ (has)

  Reference: taka$_2$ (till) aba$_1$ (now) aisii$_3$ (this) kabhii$_5$ (ever) nahii$_6$ (not) rahii$_7$ (has) hai$_8$ (been) *niiti*$_4$ (policy)

  English: Till now this never has been the *policy*

  The remaining ten sentences were reordered incorrectly. These errors are largely in clauses which deviate from the SVO order in some way – clauses with multiple subjects or objects, clauses with no object, etc.. For example, the following sentence with two subjects and objects corresponding to the verb *wearing* has not been reordered correctly.

  Input: sabhii$_1$ (all) purusha$_2$ (men) safeda$_3$ (white) evama$_4$ (and) mahilaaen$_5$ (women) kesariyaa$_6$ (saffron) vastra$_7$ (clothes) dhaarana$_8$ (wear) kiye$_9$ hue$_{10}$ (-ing) thiin$_{11}$ (were)

  Reordered: sabhii$_1$ (all) purusha$_2$ (men) safeda$_3$ (white) evama$_4$ (and) mahilaaen$_5$ (women) kesariyaa$_6$ (saffron) vastra$_7$ (clothes) dhaarana$_8$ (wear) thiin$_{11}$ (were) kiye$_9$ hue$_{10}$ (-ing)

  Reference: sabhii$_1$ (all) purusha$_2$ (men) thiin$_{11}$ (were) dhaarana$_8$ (wear) kiye$_9$ hue$_{10}$ (-

[1] The numeric subscripts in the examples indicate word positions in the input.

ing) safeda$_3$ (white) evama$_4$ (and) mahilaaen$_5$ (women) kesariyaa$_6$ (saffron)

English: All men were wearing white and the women saffron

The model possibly needs more data with patterns that deviate from the standard SOV order to learn to reorder them correctly. We could also add to the model, features pertaining to subject, object, etc.

- **Clause boundaries**: Measured on a set of 844 sentences which were marked with clause boundaries, 37 sentences (4.4 %) had reorderings that violated these boundaries. An example of such a clause-boundary violation is below:

  Input: main$_1$ (I) sarakaara$_2$ (government) kaa$_3$ (of) dhyaana$_4$ (attention) *maananiiya$_5$ (honourable) pradhaana$_6$ (prime) mantri$_7$ (minister) dvaaraa$_8$ (by) isa$_9$ (this) sabhaa$_{10}$ (house) me$_{11}$ (in) kiye$_{12}$ gaye$_{13}$ (made) isa$_{14}$ (this) vaade$_{15}$ (promise)* ki$_{16}$ ora$_{17}$ (towards) dilaanaa$_{18}$ (to bring) chaahuungaa$_{19}$ (would like)

  Reordered: main$_1$ (I) chahuungaa$_{19}$ (would like) dilaanaa$_{18}$ (to bring) kii$_{16}$ ora$_{17}$ (towards) *isa$_9$ (this) vaade$_{15}$ (promise) kiye$_{12}$ gaye$_{13}$ (made) dvaaraa$_8$ (by) maananiiya$_5$ (honourable) mantri$_7$ (minister) pradhaana$_6$ (prime)* dhyaana$_4$ (attention) kaa$_3$ (of) sarakaara$_2$ (government) *men$_{11}$ (in) isa$_{14}$ (this) sabhaa$_{10}$ (house)*

  Reference: main$_1$ (I) chahuungaa$_{19}$ (would like) dilaanaa$_{18}$ (to bring) dhyaana$_4$ (attention) kaa$_3$ (of) sarakaara$_2$ (government) *kii$_{16}$ ora$_{17}$ (towards) isa$_9$ (this) vaade$_{15}$ (promise) kiye$_{12}$ gaye$_{13}$ (made) dvaaraa$_8$ (by) maananiiya$_5$ (honourable) mantri$_7$ (minister) pradhaana$_6$ (prime) men$_{11}$ (in) isa$_9$ (this) sabhaa$_{10}$ (house)*

  English I would like to bring the attention of the government towards this promise *made by the honourable prime minister in this house*.

  Note how the italicized clause, which is kept together in the reference, is split up incorrectly in the reordered output. The proportion of such boundary violations is, however, quite low, because Hindi being a verb-final language, most clauses end with a verb and it is probably quite straightforward for the model to keep clauses separate. A clause boundary detection program should make it possible to eliminate the remaining errors.

- **Local reordering**: To estimate the short range reordering performance, we consider how often different POS bigrams in the input are reordered correctly. Here, we expect the model to reorder prepositions correctly, and to avoid any reordering that moves apart nouns and their adjectival pre-modifiers or components of compound nouns (see Section 3). Table 6 summarizes the reordering performance for these categories for a set of 280 sentences (same as the test set used in Section 5.1). Each row in Table 6 indicates the total number of correct instances for the pair, i.e., the number of instances of the pair in the reference (column titled *Total*), the number of instances that already appear in the correct order in the input (column *Input*), and the number that are ordered correctly by the reordering model (column *Reordered*). The first two rows show that adjective-noun and noun-noun (compounds) are in most cases correctly retained in the original order by the model. The final row shows that while many prepositions have been moved into their correct positions, there are still quite a few mismatches with the reference. An important reason why this happens is that nouns modified by prepositional phrases can often also be expressed as noun compounds. For example, *vidyuta (electricity) kii (of) aavashyakataaen (requirements)* in Hindi can be expressed either as "requirements of electricity" or "electricity requirements". The latter expression results in a match with the input (explaining many of the 104 correct orders in the input) and a mismatch with the model's reordering. The same problem in the training data would also adversely impact the learning of the preposition reordering rule.

POS pair	Total	Input	Reordered
adj-noun	234	192	196
noun-noun	46	44	42
prep-noun	436	104	250

Table 6: An analyis of reordering for a few POS bigrams

Language pair		BLEU	
Source	Target	Unreordered	Reordered
Hindi	English	14.7	**16.7**
Urdu	English	23.3	**24.8**
English	Hindi	20.7	**22.5**

Table 7: Translation performance without reordering (baseline) compared with performance after preordering with our reordering model.

5.5 Machine translation results

We now present experiments in incorporating the reordering model in machine translation systems. For all results presented here, we reorder the training and test data using the single best reordering based on our reordering model for each sentence. For each of the language pairs we evaluated, we trained Direct Translation Model 2 (DTM) systems (Ittycheriah and Roukos, 2007) with and without reordering and compared performance on test data. We note that the DTM system includes features that allow it to model lexicalized reordering phenomena. The reordering window size was set to +/-8 words for both the baseline and our reordered input. In our experiments, we left the word alignments fixed, i.e we reordered the existing word alignments rather than realigning the sentences after reordering. Redoing the word alignments with the reordered data could potentially give further small improvements. We note that we obtained better baseline performance using DTM systems than the standard Moses/Giza++ pipeline (e.g we obtained a BLEU of 14.9 for English to Hindi with a standard Moses/Giza++ pipeline). For all of our systems we used a combination of HMM (Vogel et al., 1996) and MaxEnt alignments (Ittycheriah and Roukos, 2005).

For our Hindi-English experiments we use a training set of roughly 250k sentences (5.5M words) consisting of the Darpa-TIDES dataset (Bojar et al., 2010) and an internal dataset from several domains but dominated by news. Our test set was roughly 1.2K sentences from the news domain with a single reference. To train our reordering model, we used roughly 6K alignments plus 17K snippets selected from MaxEnt alignments as described in Section 5.1 with bigram, ContextPOS and ContextWord features. The monolingual reordering BLEU (on the same data reported on in Section 5.3) was 54.0 for Hindi to English and 60.8 for English to Hindi.

For our Urdu-English experiments we used 70k sentences from the NIST MT-08 training corpus and used the MT-08 eval set for testing. We note that the MT-08 eval set has four references as compared to one reference for our Hindi-English test set. This largely explains the improved baseline performance for Urdu-English as compared to Hindi-English. We present averaged results for the Web and News part of the test sets. To train the reordering model we used 9K hand alignments and 11K snippets extracted from MaxEnt alignments as described in Section 5.1 with bigram, ContextPOS and ContextWord context feature. The monolingual reordering BLEU for the reordering model thus obtained (on the same data reported on in Section 5.3) was 52.7.

Table 7 shows that for Hindi to English, English to Hindi and for Urdu to English we see a gain of 1.5 - 2 BLEU points. For English $\rightarrow$ Hindi we also experimented with a system that uses rules (learned from the data using the methods described in (Visweswariah et al., 2010)) applied to a parse to reorder source side English sentences. This system had a BLEU score of 21.2, which is an improvement over the baseline, but our reordering model is better by 1.3 BLEU points.

An added benefit of our reordering model is that the decoder can be run with a smaller search space exploring only a small amount of reordering without losing accuracy but running substantially faster. Table 8 shows the variation in machine Hindi to English translation performance with varying skip size (this parameter sets the maximum number of words skipped during decoding, lower values are associated with a restricted decoder search space and increased speed).

skip	Unreordered	Reordered
2	12.2	16.7
4	13.4	16.7
8	14.7	16.4

Table 8: Translation performance with/without reordering with varying decoder search space.

6 Conclusion and future work

In this paper we presented a reordering model to reorder source language data to make it resemble the target language word order without using either a source or target parser. We showed consistent gains of up to 2 BLEU points in machine translation performance using this model to preorder training and test data. We show better performance compared to syntax based reordering rules for English to Hindi translation. Our model used only a part of speech tagger (sometimes trained with fairly small amounts of data) and a small corpus of word alignments. Considering the fact that treebanks required to build high quality parsers are costly to obtain, we think that our reordering model is a viable alternative to using syntax for reordering. We also note, that with the preordering based on our reordering model we can achieve the best BLEU scores with a much tighter search space in the decoder. Even accounting for the cost of finding the best reordering according to our model, this usually results in faster processing than if we did not have the reordering in place.

In future work we plan to explore using more data from automatic alignments, perhaps by considering a joint model for aligning and reordering. We would also like to explore doing away with the requirement of having a POS tagger, using completely unsupervised methods to class words. We currently only look at word pairs in calculating the loss function used in MIRA updates. We would like to investigate the use of other loss functions and their effect on reordering performance. We also would like to explore whether the use of scores from our reordering model directly in machine translation systems can improve performance relative to using just the single best reordering.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of ACL*, ACL-44, pages 529–536, Morristown, NJ, USA. Association for Computational Linguistics.

David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. 2005. Concorde tsp solver. In *http://www.tsp.gatech.edu/*.

Ibrahim Badr, Rabih Zbib, and James Glass. 2009. Syntactic phrase reordering for English-to-Arabic statistical machine translation. In *Proceedings of EACL*.

Ondrej Bojar, Pavel Stranak, and Daniel Zeman. 2010. Data issues in English-to-Hindi machine translation. In *LREC*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Morristown, NJ, USA. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*.

Aniket Dalal, Kumar Nagaraj, Uma Sawant, Sandeep Shelke, and Pushpak Bhattacharyya. 2007. Building feature rich pos tagger for morphologically rich languages: Experiences in Hindi. In *Proceedings of International Conference on Natural Language Processing*.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of ACL*.

D. Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Kurt Hornik and Michael Hahsler. 2009. TSP–infrastructure for the traveling salesperson problem. *Journal of Statistical Software*, 23(i02).

Sarmad Hussain. 2008. Resources for Urdu language processing. In *Proceedings of the 6th Workshop on Asian Language Resources, IJCNLP'08*.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of HLT/EMNLP*, HLT '05, pages 89–96, Stroudsburg, PA, USA. Association for Computational Linguistics.

Abraham Ittycheriah and Salim Roukos. 2007. Direct translation model 2. In *Proceedings of HLT-NAACL*, pages 57–64.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Comput. Linguist.*, 25:607–615, December.

Young-Suk Lee, Bing Zhao, and Xiaoqian Luo. 2010. Constituent reordering and syntax models for English-to-Japanese statistical machine translation. In *COLING*.

Y. Liu, Q. Liu, and S. Lin. 2006. Tree-to-String alignment template for statistical machine translation. In *Proceedings of ACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT*.

Ananthakrishnan Ramanathan, Hansraj Choudhary, Avishek Ghosh, and Pushpak Bhattacharyya. 2009. Case markers and morphology: addressing the crux of the fluency problem in English-Hindi smt. In *Proceedings of ACL-IJCNLP*.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL*.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133.

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proceedings of EMNLP*.

Karthik Visweswariah, Jiri Navratil, Jeffrey Sorensen, Vijil Chenthamarakshan, and Nandakishore Kambhatla. 2010. Syntax based reordering with automatically derived rules for improved statistical machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational Linguistics*.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP-CoNLL*.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *COLING*.

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical mt. In *Proceedings of ACL*.

Mikhail Zaslavskiy, Marc Dymetman, and Nicola Cancedda. 2009. Phrase-based statistical machine translation as a traveling salesman problem. In *Proceedings of ACL-IJCNLP*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*.

Feature-Rich Language-Independent Syntax-Based Alignment for Statistical Machine Translation

Jason Riesa[1] **Ann Irvine**[2] **Daniel Marcu**[1]

[1]Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
{riesa, marcu}@isi.edu

[2]Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
anni@jhu.edu

Abstract

We present an accurate word alignment algorithm that heavily exploits source and target-language syntax. Using a discriminative framework and an efficient bottom-up search algorithm, we train a model of hundreds of thousands of syntactic features. Our new model (1) helps us to very accurately model syntactic transformations between languages; (2) is language-independent; and (3) with automatic feature extraction, assists system developers in obtaining good word-alignment performance off-the-shelf when tackling new language pairs. We analyze the impact of our features, describe inference under the model, and demonstrate significant alignment and translation quality improvements over already-powerful baselines trained on very large corpora. We observe translation quality improvements corresponding to 1.0 and 1.3 BLEU for Arabic-English and Chinese-English, respectively.

1 Introduction

In recent years, several state-of-the-art statistical machine translation (MT) systems have incorporated both source and target syntax into the grammars that they generate and use to translate. While some tree-to-tree systems parse source and target sentences separately (Galley et al., 2006; Zollman and Venugopal, 2006; Huang and Mi, 2010), others project syntactic parses across word alignments (Li et al., 2009). In both approaches, as in largely all statistical MT, the quality of the alignments used to generate the rules of the grammar are critical to the success of the system. However, to date, most word alignment systems have not considered the same degree of syntactic information that MT systems have.

Extending unsupervised models, like the IBM models (Brown et al., 1993), generally requires changing the entire generative story. The additional complexity would likely make training such models quite expensive. Already, with ubiquitous tools like GIZA++ (Och and Ney, 2003), training accurate models on large corpora takes upwards of 5 days.

Recent work in discriminative alignment has focused on incorporating features that are unavailable or difficult to incorporate within other models, e.g. (Moore, 2005; Ittycheriah and Roukos, 2005; Liu et al., 2005; Taskar et al., 2005b; Blunsom and Cohn, 2006; Lacoste-Julien et al., 2006; Moore et al., 2006). Even more recently, motivated by the rise of syntax-based translation models, others have sought to inform alignment decisions with syntactic information (Fraser and Marcu, 2007; DeNero and Klein, 2007; May and Knight, 2007; Fossum et al., 2008; Haghighi et al., 2009; Burkett et al., 2010; Pauls and Klein, 2010; Riesa and Marcu, 2010).

Motivated by the wide modeling gap that still remains between syntax-based translation and word-alignment models, in this paper we expand on previous work in discriminative alignment, and move forward in three key areas:

1. We *heavily exploit both source and target syntax* in ways that most models can not. In addition, during training we extract and learn hundreds of thousands of features automatically, learning both the structure and parameters for the model at the same time.

2. Our model and inference support *arbitrary features*, and easily scale to millions of features.

3. Having strengthened the synchronicity between

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 497–507,
Edinburgh, Scotland, UK, July 27–31, 2011.

alignment and syntax-based translation models, we *advance state-of-the-art performance* in terms of both alignment and translation quality over already-powerful baselines on very large corpora.

2 A Feature-Rich Syntax-Aware Alignment Model

We follow Riesa and Marcu (2010) for efficient inference with arbitrary features, but do not rely upon hand-crafted syntactic patterns; rather, we extract syntactic features automatically from training data. We also introduce, in Section 5, an iterative approximate Viterbi inference procedure to deal with the asymmetry of the model. We show that this boosts both alignment and downstream translation quality even further.

The model itself is a linear combination of features, whose parameters are learned online via a structured perceptron (Collins, 2002). However, as we describe in Section 3, the features of the model are not known a priori. In what follows, we describe the search algorithm so that the reader has an understanding of the domain of locality before we begin to describe features and how they are learned.

2.1 Search Overview

We formulate the search for the best alignment as bottom-up parsing. Given a syntactic parse tree on one side of a parallel sentence, we use the structure of the tree to guide the search process. The key idea is that complex interactions between alignments are less likely to cross constituents, so we search recursively on the tree.

As an illustrative example, we point to the structure of the hypergraph search depicted in Figure 1. Here we are aligning the sentence pair:

a flag hung from the stage
台 上 挂 着 国旗
tái shàng guà zhe guóqí

The figure shows the search process for a small example with beam size k. Each black square represents a **partial alignment**. Each partial alignment at each node is ranked according to its model score. In this figure, the 1-best hypothesis at the leftmost NP node is constructed by composing the best hypothesis at its child DT and the 2nd-best hypothesis at its child NN. At the root node, we have a k-best list of full alignments.

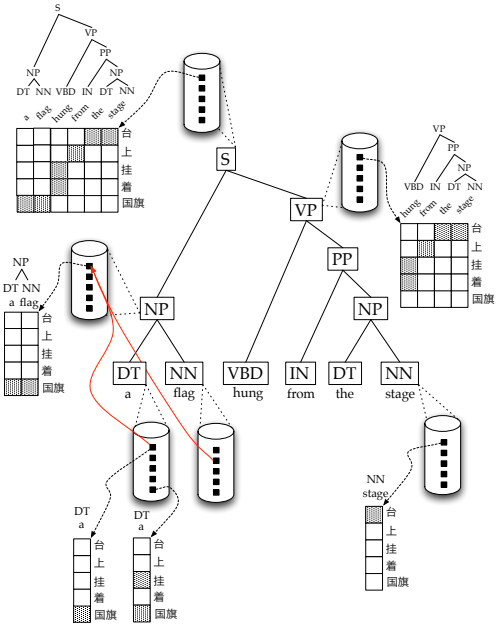


Figure 1: Approximate search through a hypergraph with beam size $k = 5$. Each black square represents a partial alignment; larger grey-shaded boxes are links in an alignment. Each partial alignment at each node is ranked according to its model score. The root node, S, contains a k-best list of full alignments.

We continue with a procedural description of the algorithm.

2.1.1 Initialization

We begin by visiting each preterminal node in sequence. We enumerate and score all one-to-one links as well as the unaligned link (aligned to null). Next, for a given preterminal node, we use cube pruning (Chiang, 2007) to find the top k one-to-two alignments, given the scores of the one-to-one links. We perform additional iterations of cube pruning to find top k sets of one-to-m links. In theory, we could increase m to the length of the foreign sentence and enumerate top k lists for each English word aligned to between 0 and all foreign words. However, in practice we set m to limit time spent here, while maintaining acceptable recall. In our experiments we set $m = 2$ for both English-Arabic and English-Chinese.

2.1.2 Combination

We continue traversing the tree bottom-up. At each nonterminal node, a k-best list of partial alignments from each of its child nodes are combined into a larger span. We use cube pruning to do this efficiently.[1] Nodes in different subtrees are processed independently of one another; i.e., for any node, alignment information at that node's sister is unavailable. For example, in Figure 1, alignment information at the leftmost NP is unavailable to us while we are constructing partial alignments at the PP. Search continues recursively up the tree, until we have reached the root node. The root node again computes the top k alignments from its children, and these comprise our final k-best list of full alignments.

In our experiments we only make use of the 1-best alignment for evaluation and translation. Previous work has shown that only shallow k-best lists of alignments may be beneficial, and that very deep k-best lists are not especially useful in improving final downstream translation grammar extraction due to rapid degradation in quality (Venugopal et al., 2008; Liu et al., 2009b); though they may have other uses.

3 Automatically Exploiting Syntactic Features for Alignment

Up to now, previous work in syntax-based alignment has largely modeled alignments based on features encoding target-side English syntactic and lexical information, but only lexical information on the source side.

However, there is much more data waiting to be exploited, and the flexible model and efficient and modular learning framework of hierarchical discriminative alignment afford us this possibility. Here, we discuss our target-side features, source-side features, and features that jointly take into account both source- and target-side information.

3.1 Target Syntax Features

Most alignment systems currently function without explicit regard to the downstream translation model. Some notable exceptions are May and Knight (2007) who generate syntactic alignments by re-aligning word-to-word alignments with a syntactic model; and Pauls and Klein (2010) who generate syntactic alignments with a synchronous ITG (Wu, 1997) approach. We depart from ITG-based models (Cherry and Lin, 2006; Haghighi et al., 2009) because of their complexity ($O(n^6)$ in the synchronous case), requiring heavy pruning or the computation of outside cost estimates (DeNero and Klein, 2010). Instead, we use linguistically motivated target-side parse trees to constrain search, as described above. These trees are output from the Berkeley parser (Petrov and Klein, 2007) and fixed at alignment time. We use these trees not only as a vehicle for search, but also for features.

A significant motivation for this work is the desire to make the connection, at alignment time, between translation rules used in decoding and the alignments that yield such translation rules. To do this, we fold the rule extraction process into the alignment search. At each step in the search process, we can extract translation rules from a given partial alignment and encode them as binary features.

Importantly, the rule extraction process itself is not directly tied to the alignment system, but rather to the downstream translation model. We can drop in any type of rule extraction we like into the alignment system, though some may generalize better than others to new data in a large corpus. This is key for supervised training conditions with relatively small amounts of annotated data.[2] In this work we focus on string-to-tree translation and the translation rule space described in (Galley et al., 2004; Galley et al., 2006).

During training and inference, we are constantly scoring partial alignments. Every time we have a partial alignment to score, we can extract all potential translation rules implied by that alignment, and encode those rules as features. In this case, we are doing two important things:

1. informing the alignment search with the rules of the translation model, and

2. modeling actual translation rules – the model parameters give us a way to quantify the relative importance of each rule.

For example, we learn that:

[1] Cube pruning is approximate when we have nonlocal combination features, and most of our features are of this type.

[2] For example, fully lexicalized phrase-based rules are less useful here than gapped phrases or hierarchical rules.

(1) Chinese VP and NP tend to be reordered around the 的 particle when translating to English.

feature	weight
NP(NP$_{\boxed{1}}$ VP$_{\boxed{2}}$) ↔ $\boxed{2}$ 的 $\boxed{1}$	1.01304

(2) When translating an Arabic NP as part of a VP, we often insert "is".

feature	weight
VP((VBZ is) NP$_{\boxed{1}}$) ↔ $\boxed{1}$	0.67252

From this process we extract and learn 326,239 lexicalized and non-lexicalized translation rule features in our Arabic-English model; 234,972 in our Chinese-English model. Those features for which a positive weight is learned tend to generalize well over the training data; negatively weighted features do not, and are generally learned from alignments with mistakes during search. See Figure 2 for additional examples of rule features learned for Arabic-English alignment.

Negative evidence Nearly 67% of the rule features we learn for Chinese-English, and 55% of the rule features we learn for Arabic-English are negatively weighted. Early experiments involved only firing indicator rule features when an extracted rule at alignment-time matched in a set of rules extracted offline from our hand-aligned data. However, coverage from such rules will always be limited; firing every rule as a feature as it is encountered during search gives us many more darts to throw. Using only rule features extracted from gold data lowers F-measure by close to 5 points.

3.2 Source Syntax Features and Joint Features

Source syntactic trees have recently been shown to be helpful in machine translation decoding (Zhang et al., 2008; Liu et al., 2009a; Chiang, 2010), but to our knowledge have not been used in alignment models other than that of Burkett et al. (2010). We parse the source side of our data using the Berkeley parser (Petrov and Klein, 2007), and encode information provided by the source syntax as features in the model in two ways: (1) as tree-distance features[3], and (2) as joint source-target syntax features.

[3]These features parameterize the intuition that if two source words align to a single target word, we prefer them to be members of the same constituent, or having a short path through the tree from one word to the other, e.g. (in, 在...中), or the first and last Chinese words in the examples in Figure 3.

Extracted Rule Feature	Weight
NP → JJ$_{[1]}$ NN$_{[2]}$ ↔ [2] [1]	1.11908
NP → JJ$_{[1]}$ NN$_{[2]}$ ↔ [1] [2]	–0.15417
PP → IN$_{[1]}$ NP$_{[2]}$ ↔ [1] [2]	1.15328
PP → IN$_{[1]}$ NP$_{[2]}$ ↔ [2] [1]	–0.65943

Figure 2: Translation rules as features extracted during Arabic-English alignment. These rules show that we learn to reorder adjectives and nouns inside noun phrases, and that prepositions before sister NPs prefer to be translated monotonically. For Chinese-English, we learn the opposite.

3.2.1 Source-Target Coordination Features

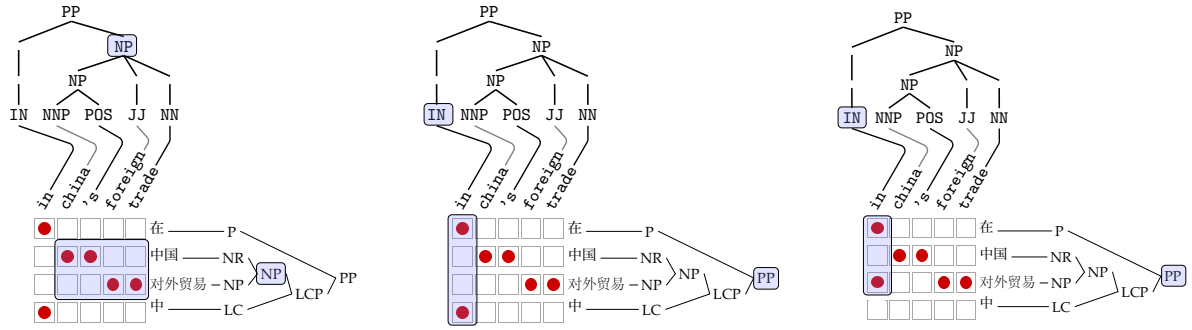
Drawing on work by Chiang (2010) in stochastically rewriting syntactic constituents across languages in a translation model, we adapt the general idea to alignment modeling. Chiang calls these features *fuzzy syntax* features; here, we simply call them *coordination* features in our adaptation for alignment, so as to avoid the implication that we are rewriting.

This feature family is a set of binary features that may fire at any nonterminal node in the tree during bottom-up search. A feature fires for each combination of two nonterminal source and target nodes s and t, respectively, that match the following conditions:

1. t is the label of the current target tree node in the bottom-up search.

2. s is the label of the source tree node of maximal depth (i.e. closest to leaf nodes) that spans all links also spanned by t.

Figure 3 shows three examples of this joint feature over source and target trees. In Figure 3a, the maximal-depth source tree node that spans every link also spanned by the shaded target tree NP



(a) Source/target tree feature firing at node NP, with value ⟨ NP ; NP ⟩. The maximal-depth source tree node that spans every link also spanned by the shaded target tree NP is also labeled NP.

(b) Source/target tree feature firing at node IN, returning value ⟨ IN ; PP ⟩.

(c) In this figure, depicting an incorrect alignment, the same feature value is fired as for the correct alignment in 3b: ⟨ IN ; PP ⟩. We need more contextual annotation to create more discriminative power.

Figure 3: Two examples of joint features over monolingual parse trees. The value of the feature depends on the shaded areas.

is also labeled NP. So, the feature returns a value of ⟨NP; NP⟩. In Figure 3b, PP is the label of the maximal-depth source tree node that spans every link also spanned by the shaded target tree IN node; the feature fires a value ⟨IN ; PP⟩. We might expect this pairing of IN with PP, or of IN with P, but we would expect to learn a penalizing parameter weight for the pairing of, say, IN with NP.

Adding more context Powerful as this feature is, it is not quite discriminating enough; it may return the same feature value for both a correct and incorrect alignment, as shown in Figure 3c. To overcome this, we introduce additional features annotated with the left-most and right-most tags in the current span. For example, in this figure, we also fire ⟨ IN ; PP(P,NP) ⟩, and learn a negative weight of −0.638 denoting a poor choice of alignment. We also find it helpful to keep the original unannotated feature as a poor-man's backoff.

Some examples Table 1 shows some of the maxmially and minimally-weighted features learned. As the more highly weighted features show, both models learn to prefer alignments that result in the coordination of similar constituent labels. For example, the Chinese model learns a very high weight for aligning sets of English words that form prepositional phrases to sets of Chinese words that also form prepositional phrases[4].

	Ara-Eng Model			Chi-Eng Model		
	eng	ara	w	eng	chi	w
[1]	SBAR	SBAR	6.40	PP	PP	10.3
[2]	S	S(CC,PU)	4.91	NP	NP	9.38
[3]	PP	PP	4.20	SBAR	VP(VV,PU)	6.97
[4]	VP	VP	3.90	NP	NP(DT,NN)	6.67
[5]	SBAR	PP	2.58	PP	PP(P,LC)	6.38
[6]	NP	S	-2.80	NP	PP	-6.82
[7]	NP	VP	-3.01	S	IP(PU,PU)	-7.44
[8]	NP	NP(NN,IN)	-4.52	PP	IP	-7.33
[9]	PP	VP	-5.13	SBAR	VP	-7.72
[10]	PP	S	-7.37	NP	IP	-7.83

Table 1: This table shows a sampling of the highest and lowest-weighted coordination features applied when scoring partial alignments at nodes in the tree. Preterminal tags inside parentheses indicate the POS tags on the left and right edge of a given constituent.

Inversely, we learn high negative weights for model features that fire for alignments that oblige the firing of features of very dissimilar nonterminal labels, and that often yield asynchronous bracketing. For example, the Arabic model learns that English words that form prepositional phrases should

[4] In Table 1, Chinese feature [1].

not align to sets of Arabic words that form entire sentences or verb phrases[5].

In total, we learn 127,932 syntactic coordination features in our Arabic-English model; 59,239 for Chinese-English.

4 Learning

We learn feature weights using a parallelized implementation of online averaged perceptron (Collins, 2002). We distribute training examples to CPUs in a cluster and essentially run several perceptron learners in parallel. We communicate and average the weight vectors of each learner according to the Iterative Parameter Mixing strategy described by McDonald et al. (2010).

Let y_i be the correct output for input x_i. Here, y_i is an alignment; x_i is a sentence pair and parse tree. At each iteration, our perceptron update is:

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{h}(y_i) - \mathbf{h}(\hat{y}) \tag{3}$$

And we define:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}(x_i)} \ell(y_i, y) + \mathbf{w} \cdot \mathbf{h}(y) \tag{4}$$

$$\ell(y_i, y) = 1 - F_1(y_i, y) \tag{5}$$

with $\mathbf{w}$ our weight vector, $\mathbf{h}(y)$ our sparse vector of feature values, $\mathcal{Y}(x_i)$ all possible outputs for input x_i, and $F_1(y_i, y)$ balanced F-measure. The loss, $\ell(y_i, y)$, is a measure of how bad it would be to guess $\hat{y}$ instead of y.

In selecting $\hat{y}$, we draw upon the loss-augmented inference literature (Tsochantaridis et al., 2004; Taskar et al., 2005a). Alignment $\hat{y}$ is the output candidate maximizing the sum of both the loss and model score. This guess appears attractive to the model, yet has low F-measure, and so is exactly the sort of output we would like to update away from.

During training, we learn both the parameters and model structure. Figure 4b shows how the size of the model grows over time. As described in Sections 2 and 3, we automatically extract and fire features given an alignment configuration and our current position in the tree. We see a steep initial growth in model size, and then begin to trail off as the number of new unique rules and negative evidence we encounter diminishes.

[5] In Table 1, Arabic features [6] and [7].

Model Selection Among models from the first iteration up to convergence, we choose the model parameters from the best performing model as measured by F-measure on a held-out development set of alignments.

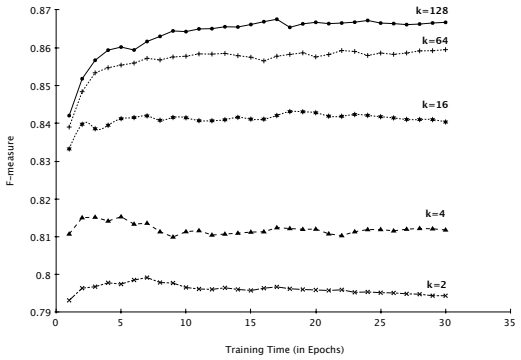
5 Iterative Approximate Viterbi Inference

Though up to now we have described features that fire during bottom-up search on the target-language tree, we can also search bottom-up on the source-language tree. The syntactic features we have described are generic enough that they will still be extractable and applicable. Because our model and inference procedure are asymmetric, a search on the source-language tree will generate alignments from a different space, and can provide a unique signal we would not otherwise have. We can use the Viterbi alignments from each model to inform the other. In the following we describe a method for simultaneously training both target-tree and source-tree models but with features to enforce agreement, somewhat similar to (Nivre and McDonald, 2008) in integrating two dependency parsing models.

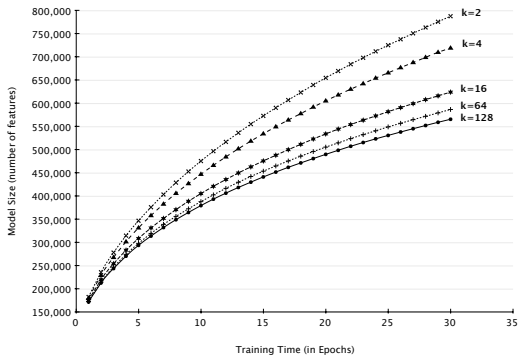
We begin by training two models, one that operates on the target tree, and one that operates on the source tree. Call the parameters learned from these models $\mathbf{w}_1^t$ and $\mathbf{w}_1^s$, respectively. Then, performing inference under these models yields alignments a_1^t and a_1^s.

In the next iteration we learn parameters $\mathbf{w}_2^t$ and $\mathbf{w}_2^s$, and introduce agreement features. In this step, during training to find $\mathbf{w}_2^t$, the target-tree model uses a_1^s to fire indicator features. These fire for any alignment link that was also present in the previous iteration's source-tree alignment, a_1^s. Analogously, when searching for the best $\mathbf{w}_2^s$, we use a_1^t to fire indicator features that fire for any alignment link also present in the previous iteration's target-tree alignment, a_1^t.

This process of using the alignment from the previous iteration's opposing tree continues until convergence, i.e. until we no longer see improvement in our 1-best source-tree and target-tree alignments. When we use these alignments for downstream translation, we symmetrize with the *grow-diag-final* heuristic, which continues to work remarkably well in practice. We also experiment with the intersection of both final alignments.



(a) Learning curves (Arabic-English): F-measure accuracy on heldout development data over time for five different beam settings, k=2, k=4, k=16, k=64 and k=128. For Arabic-English, improvements are minimal with beams larger than k=128; and for Chinese-English, with beams larger than k=256.



(b) Model size as a function of time for five different beam settings (Arabic-English): We see a steep initial growth, and then begin to trail off as the number of new unique extractable features and negative evidence we encounter diminishes. Growth rate is higher for models with narrower beams that make more mistakes.

Figure 4: Learning feature-rich alignment models. Figure 4a shows learning curves on heldout data for five different beam sizes. Figure 4b shows how the models dynamically grow over time. In Figure 4b we notice that less accurate models with narrower beams need to add more complexity in an attempt to make up for their many more mistakes.

6 Evaluation

6.1 Alignment Quality

From LDC2006E86 and LDC2006E83, we use as training data 2,280 hand-aligned sentence pairs of Arabic-English and 1,102 for Chinese-English. We measure training convergence using a held-out development set of 100 sentence pairs for each language pair, and evaluate with F-measure on a held-out test set of 184 sentences pairs for Chinese-English and 364 sentence pairs for Arabic-English. We use instances of the Berkeley parser (Petrov and Klein, 2007) trained on the English Penn Treebank, Chinese Treebank 6, and the Arabic Treebank parts 1–3; for each language, trees are fixed at alignment time using the 1-best output from each parser.

We use Model-4 symmetrized with the grow-diag-final heuristic, trained with GIZA++ as a baseline alignment model. We train two GIZA++ models on our largest available Chinese-English and Arabic-English parallel corpora. These consist of 261M and 223M English words,[6] respectively. The size of these corpora make for quite a powerful unsupervised baseline.

In training our alignment model, we use the syntactic features discussed in Section 3, plus word-based lexical features $t(e \mid f)$ and $t(f \mid e)$ used during initialization, extracted offline directly from the translation-table of GIZA++. Using these features alone results in an F-measure of 59.1 for Arabic-English, and 55.6 for Chinese-English. Our automatically extracted syntactic features and iterative inference algorithm get us the rest of the way, bringing performance up to 87.6 and 87.0, respectively.

Table 2 shows the results on our held-out 100-sentence test set. In an intrinsic evaluation on an alignment task, our F-measure scores are more than 15 points higher than the baseline for both language pairs.

6.2 Translation Quality

In evaluating downstream translation quality, we build three translation systems each for Arabic-English and Chinese-English: one with alignments from GIZA++, one with alignments from our syntactically-informed discriminative model, and one with alignments from our model with iterative inference (Section 5). For each of these systems we

[6]These counts correspond to 240M words of Chinese and 194M words of Arabic.

	Arabic-English			Chinese-English		
	F	P	R	F	P	R
GIZA++ M4 grow-diag-final	72.5	74.5	70.5	71.7	71.4	72.0
Target-tree alignments only	86.8	89.1	84.6	84.4	89.4	80.0
with Iterative Inference (grow-diag-final)	**87.6**	89.7	**85.6**	**87.0**	90.0	**84.1**
with Iterative Inference (intersection)	83.4	**93.1**	75.6	83.1	**95.4**	73.6

Table 2: F-measure, Precision, Recall for GIZA++ Model-4, and for alignments from this work. GIZA++ was trained on 223M words for Arabic-English, and 261M words for Chinese-English. We observe very large gains in accuracy of 15 points for both language pairs. Iterative inference with source and target-tree alignments yields a large effect on Chinese-English recall, and a modest improvement in Arabic-English.

align our parallel training corpora described in Section 6.1, and compute word-based lexical weighting features (Koehn et al., 2003) based on these alignments.

Because of the number of experiments involved in this research, we needed to accelerate our downstream experimental pipeline. While we align our full training corpus, we extract translation rules from a subset of our alignment training data; the quality of the translation rules extracted is still a function of the original alignment model.

We train a syntax-based string-to-tree translation model (Galley et al., 2004; Galley et al., 2006) and extract translation rules[7] using alignments produced by each system from 4.25+5.43M words for Arabic-English and 31.8+37.7M words for Chinese-English. For Arabic-English, we tune our MT system on a held-out development corpus of 1,172 parallel sentences, and test on a heldout set of 746 parallel sentences with four references each. For Chinese-English we tune our MT system on a held-out development corpus of 4,089 parallel sentences, and test on a set of 4,060 sentences with four references each. We tune the translation models for these systems with MIRA (Watanabe et al., 2007; Chiang et al., 2008). Our tuning and test corpora are drawn from the NIST 2004 and 2006 evaluation data, disjoint from our rule-extraction data. All systems used two language models; one trained on the combined English sides of our Arabic-English and Chinese-English data (480M words), and one trained on 4 billion words of English data.

MT results are shown in Table 3. We show a gain of 1.0 and 1.3 BLEU points over GIZA++ Model-4. Each is statistically significant over the baseline.

Alignment model	ara-eng **BLEU**	chi-eng **BLEU**
GIZA++ Model-4	47.6	26.2
Target-tree alignments only	48.3*	26.4+
+Iterative Inference (gdf)	48.4	27.0*
+Iterative Inference (intersection)	**48.6**+	**27.5***

Table 3: IBM BLEU scores using a syntax-based MT system. We show statistically significant gains in both language pairs over unsupervized GIZA++ Model 4 trained on very large corpora. An asterisk (*) denotes a statistically significant improvement with $p < 0.01$ over the number immediately above; a (+) denotes $p < 0.05$.

In the case of Chinese-English, we see a 1.1 BLEU gain when using iterative inference over the standard model which provides only target-tree alignments. As measured by a bootstrap resampler, this improvement is statistically significant, with $p < 0.01$.

For Arabic-English, we see a BLEU gain of 0.7 with target-tree alignments alone, and a total 1.0 BLEU gain over the baseline with iterative inference and our joint-agreement features.

We expect the limited improvement of iterative inference for Arabic-English is due to at least two factors:

1. the relative weakness of our Arabic parser, and
2. as shown in Table 2, our Arabic target-tree alignments are already quite accurate.

[7] We use the so-called *composed* rules of (Galley et al., 2006).

7 Discussion

We achieve our best downstream BLEU results when using iterative inference with source-tree and target-tree alignments, keeping the intersection.[8] These alignments have been shown to have recall in a similar neighborhood as our unsupervised baseline, but extremely high precision.

As DeNero and Klein (2010) and others have observed, the relationship between word alignment evaluation metrics and BLEU score remains tenuous at best. While we are able to induce some of the most accurate alignments we have seen to date, it remains unclear, given our gold hand-aligned data, whether we are optimizing for the right function ultimately for the translation task. Related metrics, like Rule F-measure (Fossum et al., 2008) and Translation Unit Error Rate (Søgaard and Kuhn, 2009), are still functions of a given gold alignment. If the gold alignment is not ideally annotated for the translation task, it matters little what our alignment evaluation metric is.

Why do grow-diag-final alignments (for our system) not perform as well? We believe the answer lies in the fact that these alignments *too closely* resemble the gold alignments with word-alignment annotation standards[9] that do not handle function words ideally for the translation task. Indeed, Hermjakob (2009) reports improved BLEU with a hand-modified gold standard.

Interestingly, the places in which our source-tree and target-tree alignments most often disagree is in the alignment of function words with no clear translation in the opposite language. For example, English *the* has no translation in Chinese. Our intersection alignments generally leave *the* unaligned to Chinese words, whereas in our gold alignments *the* is generally aligned to the same word as the head of the NP in which it appears.[10]

We see our best translation performance with our intersection alignments because we believe it largely leaves untranslated words and words without clear translations in the opposite language unaligned; we believe this may be the right thing to do.[11] Continuing with the *the* example, our translation model learns to insert words like *the* where appropriate, and such insertion rules are validated by the language model. We learn with good coverage accurate high-precision translation rules for content words, and general insertion rules for words like *the*, instead of learning two unique lexicalized rules for a given content word, one with and one without *the*. In this way, we are learning a more general grammar that explains the data.

8 Conclusion

In this work we are closing the gap between translation and alignment models in terms of syntactic sophistication. We have (1) shown how to efficiently extract hundreds of thousands of language-independent syntactic features useful for alignment, (2) given a detailed analysis of the types of linguistic phenomena these varied features generalize, and (3) report significant gains not only on alignment quality but also on downstream machine translation quality (1.0+ BLEU) over very strong baselines across diverse language pairs.

We have also hinted at roadblocks to improved discriminative alignment modeling for translation. We expect that an accurate discriminative word alignment system, such as the one presented here, in conjunction with better annotation standards for alignment will take us even farther beyond the advancements in translation quality shown here.

Acknowledgements

The authors would like to thank David Chiang, Steve DeNeefe, Liang Huang, Kevin Knight, Jonathan May, and the anonymous reviewers for their thoughtful comments. This work was supported in part by NSF IIS-0908532, DARPA contract HR0011-06-C-0022 under subcontract to BBN Technologies, and a USC CREATE Fellowship to the first author.

[8] Intersection symmetrization does not help GIZA++ because the resulting recall is so low as to severely limit the usefulness of direct translation rule extraction with such alignments (49.7 Recall for Chi-Eng; 47.2 Recall for Ara-Eng).

[9] We refer to those used for data used in this work, LDC2006E86 and LDC2006E93, as well as the standards for later hand-aligned data developed for the GALE program.

[10] E.g., ⟨(the country , 国家)⟩; but not, ⟨(the, ∅); (country, 国家)⟩

[11] Naively leaving all function words unaligned is likely suboptimal, as many have seem to have direct translations in some contexts; cf. (of, من) and (of, 的).

References

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of COLING-ACL*, pages 65–72, Sydney, Australia.

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

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Proceedings of NAACL HLT 2010*, pages 127–135, Los Angeles, CA. USA.

Colin Cherry and Dekang Lin. 2006. Soft syntactic constraints for word alignment through discriminative training. In *Proceedings of COLING/ACL*, pages 105–112, Sydney, Australia. Association for Computational Linguistics.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 224–233, Honolulu, HI. USA.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the ACL*, pages 1443–1452, Uppsala, Sweden.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8, Philadelphia, PA. USA.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of the 45th annual meeting of the ACL*, pages 17–24, Prague, Czech Republic.

John DeNero and Dan Klein. 2010. Discriminative modeling of extraction sets for machine translation. In *Proceedings of NAACL HLT 2010*, pages 1453–1463, Los Angeles, CA. USA.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using syntax to improve word alignment for syntax-based statistical machine translation. In *Proceedings of ACL MT Workshop*, pages 44–52, Honolulu, HI. USA.

Alexander Fraser and Daniel Marcu. 2007. Getting the structure right for word alignment: LEAF. In *Proceedings of EMNLP-CoNLL*, pages 51–60, Prague, Czech Republic.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL*, pages 961–968, Sydney, Australia. Association for Computational Linguistics.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the ACL and IJCNLP*, pages 923–931, Singapore, August.

Ulf Hermjakob. 2009. Improved word alignment with statistics and linguistic heuristics. In *Proceedings of EMNLP*, pages 229–237, Singapore.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of EMNLP 2010*, pages 273–283, Boston, MA. USA.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of HLT-EMNLP*, pages 89–96, Vancouver, Canada.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133, Edmonton, Canada.

Simon Lacoste-Julien, Dan Klein, Ben Taskar, and Michael Jordan. 2006. Word alignment via quadratic assignment. In *Proceedings of HLT-NAACL*, pages 112–119, New York, NY. USA.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of the 43rd annual meeting of the ACL*, pages 459–466, Ann Arbor, MI.

Yang Liu, Yajuan Lü, and Qun Liu. 2009a. Improving tree-to-tree translation with packed forests. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 558–566, Singapore.

Yang Liu, Tian Xia, Xinyan Xiao, and Qun Liu. 2009b. Weighted alignment matrices for statistical machine translation. In *Proceedings of EMNLP*, pages 1017–1026, Singapore.

Jonathan May and Kevin Knight. 2007. Syntactic realignment models for machine translation. In *Proceedings of EMNLP*, pages 360–368, Prague, Czech Republic.

Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured perceptron. In *Proceedings of NAACL HLT*, pages 456–464, Los Angeles, CA. USA.

Robert C. Moore, Wen-Tau Yih, and Andreas Bode. 2006. Improved discriminative bilingual word alignment. In *Proceedings of COLING-ACL*, pages 513–520, Sydney, Australia.

Robert C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of HLT-EMNLP*, pages 81–88, Vancouver, Canada.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the ACL*, pages 950–958, Columbus, OH. USA.

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

Adam Pauls and Dan Klein. 2010. Unsupervised syntactic alignment with inversion transduction grammars. In *Proceedings of NAACL HLT 2010*, pages 118–126, Los Angeles, CA. USA.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT*, pages 404–411, Rochester, NY. USA.

Jason Riesa and Daniel Marcu. 2010. Hierarchical search for word alignment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 157–166, Uppsala, Sweden.

Anders Søgaard and Jonas Kuhn. 2009. Empirical lower bounds on alignment error rates in syntax-based machine translation. In *SSST '09: Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 19–27. Association for Computational Linguistics.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005a. Learning structured prediction models: A large margin approach. In *Proceedings of ICML*, pages 896–903, Bonn, Germany.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005b. A discriminative matching approach to word alignment. In *Proceedings of HLT-EMNLP*, pages 73–80, Vancouver, Canada.

Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of ICML*, Banff, AB. Canada.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2008. Wider pipelines: *N*-best alignments and parses in MT training. In *Proceedings of AMTA*, pages 192–201, Honolulu, HI. USA.

Taro Watanabe, Jun Suzuki, Hajime Tsukuda, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of EMNLP*, pages 764–773.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL-08: HLT*, pages 559–567, Columbus, OH. USA.

Andreas Zollman and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *NAACL 2006 Workshop on Statistical Machine Translation*, pages 138–141, Rochester, NY. USA.

Efficient retrieval of tree translation examples for Syntax-Based Machine Translation

Fabien Cromieres
Graduate School of Informatics
Kyoto University
Kyoto, Japan
fabien@nlp.kuee.kyoto-u.ac.jp

Sadao Kurohashi
Graduate School of Informatics
Kyoto University
Kyoto, Japan
kuro@i.kyoto-u.ac.jp

Abstract

We propose an algorithm allowing to efficiently retrieve example treelets in a parsed tree database in order to allow on-the-fly extraction of syntactic translation rules. We also propose improvements of this algorithm allowing several kinds of flexible matchings.

1 Introduction

The popular Example-Based (EBMT) and Statistical Machine Translation (SMT) paradigms make use of the translation examples provided by a parallel bilingual corpus to produce new translations. Most of these translation systems process the example data in a similar way: The parallel sentences are first word-aligned. Then, translation rules are extracted from these aligned sentences. Finally, the translation rules are used in a decoding step to translate sentences. We use the term translation rule in a very broad sense here, as it may refer to substring pairs as in (Koehn et al., 2003), synchronous grammar rules as in (Chiang, 2007) or treelet pairs as in (Quirk et al., 2005; Nakazawa and Kurohashi, 2008).

As the size of bilingual corpus grow larger, the number of translation rules to be stored can easily become unmanageable. As a solution to this problem in the context of phrase-based Machine Translation, (Callison-Burch et al., 2005) proposed to pre-align the example corpora, but delay the rule extraction to the decoding stage. They showed that using Suffix Arrays, it was possible to efficiently retrieve all sentences containing substrings of the sentence to be translated, and thus extract the needed translation rules on-the-fly. (Lopez, 2007) proposed an extension of this method for retrieving discontinuous substrings, making it suitable for systems such as (Chiang, 2007).

In this paper, we propose a method to apply the same idea to Syntax-Based SMT and EBMT (Quirk et al., 2005; Mi et al., 2008; Nakazawa and Kurohashi, 2008). Since Syntax-Based systems usually work with the parse trees of the source-side sentences, we will need to be able to retrieve efficiently examples trees from fragments (treelets) of the parse tree of the sentence we want to translate. We will also propose extensions of this method allowing more flexible matchings.

2 Overview of the method

2.1 Treelet retrieval

We first formalize the setting of this chapter by providing some definitions.

Definition 2.1 (Treelets). A *treelet* is a connected subgraph of a tree. A treelet T_1 is a *subtreelet* of another treelet T_2 if T_1 is itself a connected subgraph of T_2. We note $|T|$ the number of nodes in a treelet. If $|T| = 1$, T is called an *elementary treelet*. A *linear treelet* is a treelet whose nodes have at most 1 child. A *subtree* rooted at node n of a tree $\mathcal{T}$ is a treelet containing all nodes descendants of n.

Definition 2.2 (Sub- and Supertreelets). If T_1 is a subtreelet of T_2 and $|T_1| = |T_2| - 1$, we call T_1 an *immediate subtreelet* of T_2. Reciprocally, T_2 is an *(immediate) supertreelet* of T_1. Furthermore, if T_2 and T_1 are rooted at the same node in the original tree, we say that T_2 is a *descending supertreelet* of T_1. Otherwise it is an *ascending supertreelet* of T_1.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 508–518,
Edinburgh, Scotland, UK, July 27–31, 2011.

In treelet retrieval, we are given a certain treelet type and want to find all of the tokens of this type in the database $\mathcal{DB}$. Each token of a given treelet type will be identified by a mapping from the node of the treelet type to the nodes of the treelet token in the database.

Definition 2.3 (Matching). Given a treelet T and a tree database $\mathcal{DB}$, a matching of T in $\mathcal{DB}$ is a function M that associate the treelet T to a tree $\mathcal{T}$ in $\mathcal{DB}$ and every node of T to nodes of $\mathcal{T}$ in such a way that: $\forall n \in T, label(M(n)) = label(n)$ and $\forall (n_1, n_2) \in T$ s.t n_2 is a child of n_1, $M(n_2)$ is a child of $M(n_1)$.

In the common case where the siblings of a tree are ordered, a matching must satisfy the additional restriction: $\forall n_1, n_2 \in T, n_1 <_s n_1 \Leftrightarrow M(n_1) <_s M(n_1)$, where $<_s$ is the partial order relation between nodes meaning "is a sibling and to the left of"

We note $occ(T)$ (for "occurrences of T ") the set of all possible matchings from T to $\mathcal{DB}$. We will call *computing* T the task of finding $occ(T)$. If $|occ(T)| = 0$, we call T an *empty treelet*. *Computing a query tree* $\mathcal{T}_Q$ means computing all of its treelets.

Definition 2.4 (Notations). Although treelets are themselves trees, we will use the word *treelet* to emphasize they are a subpart of a bigger tree. We will note T a treelet, and $\mathcal{T}$ a tree. $\mathcal{T}_Q$ is the query tree we want to compute. $\mathcal{DB}$ will refer to the set of trees in our database. We will use a bracket notation to describe trees or treelets. Thus "a(b c d(e))" is the tree at the bottom of figure 2.

2.2 General approach

There exists already a large body of research about tree pattern matching (Dubiner et al., 1994; Bruno et al., 2002). However, our problem is quite different from finding the tokens of a given treelet in a database. We actually want to find all the tokens of all of the treelets of a given query tree. The query tree itself is unlikely to appear in full even once in the database. In this respect, our approach will have many similarities with (Callison-Burch et al., 2005) and (Lopez, 2007), and can be seen as an extension of these works.

The basis of the method in (Lopez, 2007) is to look for the occurrences of continuous substrings using a Suffix Array, and then intersect them to find the occurrences of discontinuous substrings. We will have a similar approach with two variants. The first variant consists in using an adaptation of the concept of suffix arrays to trees, which we will call Path-To-Root Arrays (section 3.4), that allows us to find efficiently the set of occurrences of a linear treelet. Occurrences of non-linear treelets can then be computed by intersection. The second variant is to use an inverted index (section 3.5). Then the occurrences of all treelets, even the linear treelets, are computed by intersection.

The main additional difficulty in considering trees instead of strings is that while a string has a quadratic number of continuous substrings, a tree has in general an exponential number of treelets (eg. several trillion for the dependency tree of a 70 words sentence). There is also an exponential number of discontinuous substrings, but (Lopez, 2007) only consider substrings of bounded size, limiting this problem. We will not try to bound the size of treelets retrieved. It is therefore crucial to avoid computing the occurrences of treelets that have no occurrences in the database, and also to eliminate as much redundant calculation as is possible.

Lopez proposes to use Prefix Trees for avoiding any redundant or useless computation. We will use a similar idea but with an hypergraph that we will call "computation hypergraph" (section 3.2). This hypergraph will not only fit the same role as the Prefix Tree of (Lopez, 2007), but also will allow us to easily implement different search strategies for flexible search (section 6).

2.3 Representing positions

Whether we use a Path-to-Root Array or an inverted index, we will need a compact way to represent the position of a node in a tree. It is straightforward to define such a position for strings, but slightly less for trees. Especially, if we consider ordered trees, we will want to be able to compare the relative location of the nodes by comparing their positions.

The simplest possibility is to use an integer corresponding to the rank of the node in a in-order depth-first traversal of the tree. It is then easy, for two nodes b and c, children of a parent node a, to check if b is on the left of c, or on the left of a, for example.

A more advanced possibility is to use a representation inspired from (Zhang et al., 2001), in which

the position of a node is a tuple consisting of its rank in a preorder (ie. children last) and a postorder (children first) depth-first traversal, and of its distance to the root. This allows to test easily whether a node is an ancestor of another, and their distance to each other. This allows in turn to compute by intersection the occurrences of discontinuous treelets, much like what is done in (Lopez, 2007) for discontinuous strings. This is discussed in section 7.2.

3 Computing treelets incrementally

We describe here in more details how the treelets can be efficiently computed incrementally.

3.1 Dependence of treelet computation

Let us first define how it is possible to compute a treelet from two of its subtreelets. Let us consider a treelet T and two treelets T_1 and T_2 such that $T = T_1 \cup T_2$, where, in the equality and the union, the treelet are seen as the set of their nodes. There are two possibilities. If $T_1 \cap T_2 = \emptyset$, then the root of T_1 is a child of a node of T_2 or vice-versa. We then say that T_1 and T_2 form a disjoint coverage (abbreviated as D-coverage) of T. If $T_1 \cap T_2 \neq \emptyset$, we will say that T_1 and T_2 form an overlapping coverage (abbreviated as O-coverage) of T.

Given two treelets T_1 and T_2 forming a coverage of T, we can compute $occ(T)$ from $occ(T_1)$ and $occ(T_2)$ by combining their matchings.

Definition 3.1 (compatibility for O-coverage). Let T be a treelet of $\mathcal{T}_Q$. Let T_1 and T_2 be 2 treelets forming a O-coverage of T. Let $M_1 \in occ(T_1)$ and $M_2 \in occ(T_2)$. M_1 and M_2 are compatible if and only if $M_1|_{T_1 \cap T_2} = M_2|_{T_1 \cap T_2}$ and $\mathcal{I}(M_1|_{T_1 \setminus T_2}) \cap \mathcal{I}(M_2|_{T_2 \setminus T_1}) = \emptyset$.

In the definition above, $|_S$ is the restriction of a function to a set S and $\mathcal{I}$ is the image set of a function.

If the children of a tree are ordered, we must add the additional restriction: $\forall (n_1, n_2) \in (T_1 \setminus T_2) \times (T_2 \setminus T_1), n_1 <_s n_2 \Leftrightarrow M_1(n_1) <_s M_2(n_2)$.

Definition 3.2 (compatibility for D-coverage). Let T_1 and T_2 be 2 treelets forming a D-coverage of T. Let's suppose that the root n_2 of T_2 is a child of node n_1 of T_1. Let $M_1 \in occ(T_1)$ and $M_2 \in occ(T_2)$. M_1 and M_2 are compatible if and only if $M_2(n_2)$ is a child of $M_1(n_1)$.

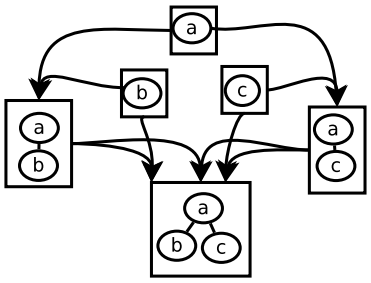


Figure 1: A computing hypergraph for "a(b c)".

Definition 3.3 (intersection ($\otimes$) operation). If two matchings are compatible, we can form their union, which is defined as $(M_1 \cup M_2)(n) = M_1(n)$ if $n \in T_1$ and $M_2(n)$ else. We note $occ(T_1) \otimes occ(T_2) = \{M_1 \cup M_2 \mid M_1 \in occ(T_1), M_2 \in occ(T_2)$ and M_1 is compatible with $M_2\ \}$. Then,we have the property: $occ(T) = occ(T_1) \otimes occ(T_2)$

In practice, the intersection operation will be implemented using merge and binary merge algorithms (Baeza-Yates and Salinger, 2005), following (Lopez, 2007).

3.2 The computation hypergraph

We have seen that it is possible to compute $occ(T)$ from two subtreelets forming a coverage of T. This can be represented by a hypergraph in which nodes are all the treelets of a given query tree, and every pair of overlapping or adjacent treelet is linked by an hyperedge to their union treelet. Whenever we have computed two starting points of an hyper-edge, we can compute its destination treelet. An example of a small computation hypergraph is described in figure 1.

It is very convenient to represent the incremental computation of the treelets as a traversal of this hypergraph. First because it contributes to avoid redundant computations: each treelet is computed only once, even if it is used to compute several other treelets. Also, if a query tree contains two distinct but identical treelets, only one computation will be done, provided the two treelets are represented by the same node in the hypergraph. The hypergraph also allows us to avoid computing empty treelets, as we describe in next section. This hypergraph therefore has the same role for us as the prefix tree used

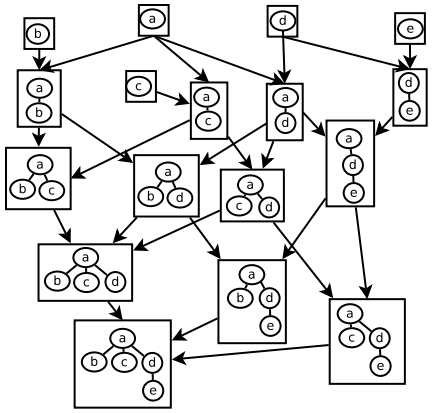


Figure 2: Inclusion DAG for the tree $a(bcd(e))$

in (Lopez, 2007). Of course, the hypergraph is generated on-the-fly during the traversal.

Furthermore, different traversals will define different computation strategies, and we will be able to use some more advanced graph exploration methods in section 6.

3.3 The Immediate Inclusion DAG

In many cases (but not always: see section 4.3), the most optimal computation strategy should be to always compute a treelet from two of its immediate subtreelets. This is because the computation time will be proportional to the size of the smallest occurrence set of the two treelets, and thus the "cheapest" subtreelet is always one of the immediate subtreelets. With this computation strategy, we can replace the general computation hypergraph by a DAG (Directed Acyclic Graph) in which every treelet point to its immediate supertreelets. An example is given on figure 2. We will call this DAG the (Immediate) Inclusion DAG.

Traversals of the Inclusion DAG should be pruned when an empty treelet is found, since all of its supertreelets will also be empty. The algorithm 1 provide a general traversal of the DAG avoiding to compute as many empty treelets as possible. It uses a queue $\mathcal{D}$ of discovered treelets, and a data-structure $\mathcal{C}$ that associate a treelet to those of its subtreelets that have been already computed. Once a treelet T has been computed and is found to be non empty, we discover its immediate supertreelets $T^{S1}, T^{S2}, \ldots$ (if they have not been discovered already) and add T to $\mathcal{C}$ (T^{S1}), $\mathcal{C}$ (T^{S2}), ... The operation min($\mathcal{C}$ (T)) retrieve the 2 subtreelets from $\mathcal{C}$ (T) that have the least occurrences. If one of them is empty, we can directly conclude that T is empty. No treelet whose all immediate subtreelets are empty is ever put in the discovered queue, which allows us to prune most of the empty treelets of the Inclusion DAG.

Algorithm 1: Generic DAG traversal

```
1  Add the set of precomputed treelets to 𝒟;
2  while ∃T ∈ 𝒟 s.t T ∈ precomputed or |𝒞(T)| > 2
   do
3      pop T from 𝒟;
4      if T in precomputed then
5          occ(T) ← precomputed[T];
6      else
7          T1,T2=min(𝒞 (T));
8          if |occ(T1)| = 0 then
9              occ(T) ← ∅;
10         else
11             occ(T) ← occ(T1) ⊗ occ(T2);
12     for T^S ∈ supertree(T) do
13         if occ(T^S) = undef then
14             Add T to 𝒞(T^S);
15             if |occ(T)| > 0 and T^S ∉ 𝒟 then
16                 Add T^S to 𝒟;
```

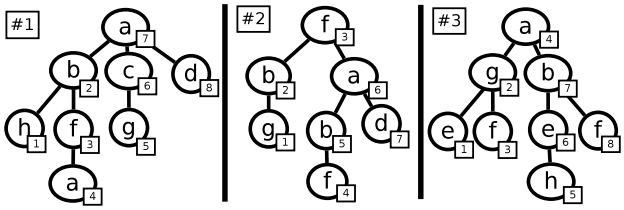
A treelet in the inclusion DAG can be computed as soon as two of its antecedents have been computed. To start the computation (or rather, "seed" it), it is necessary to know the occurrences of treelet of smaller size. In the following sections 3.4 and 3.5, we describe two methods for efficiently obtaining the set of occurrences of some initial treelets.

3.4 Path-to-Root Array

We present here a method to compute very efficiently $occ(T)$ when T is linear. This method is similar to the use of Suffix Arrays (Manber and Myers, 1990) to find the occurrences of continuous substrings in a text.

Definition 3.4 (Paths-to-Root Array). Given a labeled tree $\mathcal{T}$ and a node $n \in \mathcal{T}$, the path-to-root of n is the sequence of labels from n to the root. The *Paths-to-Root Array* of a set of trees $\mathcal{DB}$ is the lexicographically sorted list of the Path-to-Roots of every node in $\mathcal{DB}$.

Just as with suffixes, a path-to-root can be represented compactly by a pointer to its starting node in $\mathcal{DB}$. We then need to keep the database $\mathcal{DB}$ in



	Pos	PtR		Pos	PtR		Pos	PtR
1	3:4	a	8	2:2	bf	15	1:3	fba
2	1:7	a	9	1:6	ca	16	3:3	fga
3	2:6	af	10	1:8	da	17	3:2	ga
4	1:4	afba	11	2:7	daf	18	2:1	gbf
5	3:7	ba	12	3:1	ega	19	1:5	gca
6	1:2	ba	13	2:3	f	20	1:1	hba
7	2:5	baf	14	3:8	fba	21	3:5	heba

Figure 3: Path To Root Array for a set of three trees. "Pos." is the position of the starting point of a given path-to-root (noted as indexOfTree:positionInTree), and PtR is the sequence of labels on this path-to-root. The path-to-root are sorted in lexicographic order. We can find the set of occurrences of any linear treelet with a binary search. For example, the treelet a(b) corresponds to the label sequence "ba". With a binary search, we find that the path-to-root starting with "ba" are between indexes 5 and 7. The corresponding occurrences are then 3:7, 1:2 and 2:5.

memory to retrieve efficiently the pointed-to path-to-root. Once the Path-to-Root Array is built, for a linear treelet T, we can find its occurrences by a binary search of the first and last path-to-root starting with the labels of T. See figure 3 for an example.

Memory cost is quite manageable, since we only need 10 bytes per nodes in total. 5 bytes per pointer in the array (tree id: 4 bytes, start position: 1 byte), and 5 bytes per nodes to store the database in memory (label id:4 bytes, parent position: 1 byte).

All the optimization tricks proposed in (Lopez, 2007) for Suffix Arrays can be used here, especially the optimization proposed in (Zhang and Vogel, 2005).

3.5 Inverted Index and Precomputation

Instead of a Path-to-Root array, one can simply use an inverted index. The inverted index associates with every label the set of its occurrences, each occurrences being represented by a tuple containing the index of the tree, the position of the label in the tree, and the position of the parent of the label in the tree. Knowing the position of the parent will allow to compute treelets of size 2 by intersection (D-coverage). This is less effective than the Path-To-Root Array approach, but open the possibilities for the flexible search discussed in section 6.

Taking the idea further, we can actually consider the possibility of precomputing treelets of size greater than 1, especially if they appear frequently in the corpus.

4 Practical implementation of the traversal

4.1 Postorder traversal

The way we choose the treelet to be popped out on line 3 of algorithm 1 will define different computation strategies. For concreteness, we describe now a more specific traversal. We will process treelets in an order depending on their root node. More precisely, we consider the nodes of the query tree in the order given by a depth-first postorder traversal of the query tree. This way, when a treelet rooted at n is processed, all of the treelets rooted at a descendant of n have already been processed.

We can suppose that every processed treelet is assigned an index that we note $\#T$. This allows a convenient recursive representation of treelets.

Definition 4.1 (Recursive representation). Let T be a treelet rooted at node n of $\mathcal{T}_Q$. We note n_i the i^{th} child of n in $\mathcal{T}_Q$. For all i, t_i is the subtree of T rooted at n_i. We note $t_i = \emptyset$ and $\#t_i = 0$ if T does not contain n_i. The recursive representation of T is then: $[n, (\#t_1, \#t_2, \ldots, \#t_m)]$. We note T^i the value $\#t_i$.

For example, if $\mathcal{T}_Q$ ="a(b c d(e))" and the treelets "b" and "d(e)" have been assigned the indexes 2 and 4, the recursive representation of the treelet "a(b d(e))" would be [a,(2,0,4)].

Algorithm 2 describes this "postorder traversal". $\mathcal{D}_{Node}$ is a priority queue containing the treelets rooted at $Node$ discovered so far. The priority queue pop out the smallest treelets first. Line 14 maintain a list $\mathcal{L}$ of processed treelets and assign the index of T in $\mathcal{L}$ to $\#T$. Line 22 keeps track of the non-empty immediate supertreelets of every treelet through a dictionary $\mathcal{S}$. This is used in the procedure *compute-supertreelets* (algorithm 3) to generate the immediate supertreelets of a treelet T given its recursive representation. In this procedure, line 6 produces the

Algorithm 2: DAG traversal by query-tree postorder

```
1  for Node in postorder-traversal(query-tree) do
2      T_elem = [Node, (0, 0, .., 0)];
3      D_Node ← T_elem;
4      while |D_Node| > 0 do
5          T=pop-first(D_Node);
6          if T in precomputed then
7              occ(T) ← precomputed[Node.label];
8          else
9              T_1, T_2=min(C (t));
10             if |occ(T_1)| = 0 then
11                 occ(T) ← ∅;
12             else
13                 occ(T) ← occ(T_1) ⊗ occ(T_2);
14         Append T to L;
15         #T ← |L|;
16         for T^S in compute-supertree(T, #T) do
17             Add T to C(T^S);
18             if |occ(T)| > 0 then
19                 if T^S ∉ D_Node and
                   root(T^S)=Node then
20                     Add T^S to D;
21                 for #t in C (T) do
22                     Add #T to S(#t);
```

Algorithm 3: compute-supertrees

```
   Input: T,#T
   Output: lst: list of immediate supertreelets of T
1  m ← |root(T)|;
2  for i in 1 . . . m do
3      for #T^S in S(#T^i) do
4          if root(#T^S) ≠ root(T) then
5              T_new ← [root(T), T^0, ..#T', . . . , T^m];
6              Append T_new to lst;
7  T_new ← [parent(root(T)), (0, . . . , #T, . . . , 0)];
8  Append T_new to lst;
```

descending supertreelets, and line 8 produces the ascending supertreelet. Figure 4 describes the content of all these data structures for a simple run of the algorithm.

This postorder traversal has several advantages. A treelet is only processed once all of its immediate supertreelets have been computed, which is optimal to reduce the cost of the $\otimes$ operation. The way the procedure *compute-supertreelets* discover supertreelets from the info in $\mathcal{S}$ has also several benefit. One is that, by not adding empty treelets (line 18) to $\mathcal{S}$, we naturally prevent the discovery of larger empty treelets. Similarly, in the next section, we will be able to prevent the discovery of non-maximal treelets by modifying $\mathcal{S}$. Modifications of *compute-supertreelets* will also allow different kind of retrieval in section 6.

4.2 Pruning non-maximal treelets

We now try to address another aspect of the overwhelming number of potential treelets in a query tree. As we said, in most practical cases, most of the larger treelets in a query tree will be empty. Still, it is possible that some tree exactly identical to the query tree (or some tree having a very large treelet in common with the query tree) do exist in the database. This case is obviously a best case for translation, but unfortunately could be a worst-case for our algorithm, as it means that all of the (possibly trillions of) treelets of the query tree will be computed.

To solve this issue, we try to consider a concept analogous to that of maximal substring, or substring class, found in Suffix Trees and Suffix Arrays (Yamamoto and Church, 2001). The idea is that in most cases where a query tree is "full" (that is all of its treelets are not empty), most of the larger treelets will share the same occurrences (in the database trees that are very similar to the query tree). We formalize this as follow:

Definition 4.2 (domination and maximal treelets). Let T_1 be a subtreelet of T_2. If for every matching M_1 of $occ(T_1)$, there exist a matching M_2 of $occ(T_2)$ such that $M_2|_{T1} = M_1$, we say that T_1 is dominated by T_2. A treelet is maximal if it is not dominated by any other treelet.

If T_1 is dominated by T_2, it means that all occurrences of T_1 are actually part of an occurrence of T_2. We will therefore be, in general, more interested by the larger treelet T_2 and can prune as many non-maximal treelets as we want in the traversal. The key point is that the algorithm has to avoid discovering most non maximal treelets.The algorithm 2 can easily be modified to do this. We will use the following property.

Property 4.1. Given k treelets $T_1 \dots T_k$ with k distinct roots, all the roots being children of a same node n. We note $n(T_1 \dots T_k)$ the treelet whose root is n, and for which the k subtrees rooted at the k

T	d	e	b	b(d) *[Empty]*	b(e)	b(d e) *[Empty]*	c	a	a(b)	a(b(e))	a(c)	a(b c)	a(b(e) c)
#	1	2	3	4	5	6	7	8	9	10	11	12	13
R	d	e	b(..)	b(1.)	b(.2)	b(1 2)	c	a(..)	a(3.)	a(5.)	a(.7)	a(3 7)	a(5 7)
$\mathcal{C}$	-	-	-	1,3	2,3	4,5	-	-	8,3	5,9	7,8	9,11	10,12
$\mathcal{S}$	-	-	5	-	-	-	-	9,11	10,12	13	12	13	-

Figure 4: A run of the algorithm 2, for the query tree a(b(d e) c). The row "T" represents the treelets in the order they are discovered. The row "#" is the index #T, and the row "R" is the recursive representation of the treelet. Also represented are the content of $\mathcal{C}$ and $\mathcal{S}$ at the end of the computation. When a treelet is poped out of $\mathcal{D}_{Node}$, occ(T) is computed from the treelets listed in $\mathcal{C}$ (T). If occ(T) is not empty, the entries of the immediate subtreelets of T in $\mathcal{S}$ are updated with #T. We suppose here that $|occ($ b(d))|=0. Then, b(d e) is marked as empty and neither b(d) nor b(d e) are added to the entries of their subtreelets in $\mathcal{S}$. This way, when considering treelets rooted at the upper node "a", the algorithm will not discover any of the treelets containing b(d).

children of n are $T_1 \dots T_k$. Let us further suppose that for all i, T_i is dominated by a descending supertreelet T_i^d (with the possibility that $T_i = T_i^d$). Then $n(T_1 \dots T_k)$ is dominated by $n(T_1^d \dots T_k^d)$. For example, if b(c) is dominated by b(c d), then a(b(c) e) will be dominated by a(b(c d) e).

In algorithm 2, after processing each node, we proceed to a cleaning of the $\mathcal{S}$ dictionary in the following way: for every treelet T (considering the treelets by increasing size) that is dominated by one of its supertreelets $T^S \in \mathcal{S}(T)$ and for every subtreelet T' of T such that $T \in \mathcal{S}(T')$, we replace T by T^S in $\mathcal{S}(T')$. The procedure *compute-supertreelets*, when called during the processing of the parent node, will thus skip all of the treelets that are "trivially" dominated according to property 4.1.

Let's note that testing for the domination of a treelet T by one of its supertrelets T^S is not a matter of just testing if $|occ(T)| = |occ(T^S)|$, as would be the case with substring: a treelet can have less occurrences than one of its supertreelets (eg. b(a) has more occurrences than b in b(a a)). An efficient way is to first check that the two treelets occurs in the same number of sentences, then confirm this with a systematic check of the definition.

4.3 The case of constituent trees

We have focused our experiments on dependency trees, but the method can be applied to any tree. However, the computations strategies we have used might not be optimal for all kind of trees. In a dependency tree, nodes are labeled by words and most non-elementary treelets have a small number of occurrences. In a constituent tree, many treelets containing only internal nodes have a high frequency and will be expensive to compute.

If we have enough memory, we can solve this by precomputing the most common (and therefore expensive) treelets.

However, it is usually not very interesting to retrieve all the occurrences of treelets such as "NP(Det NN)" in the context of a MT system. Such very common pattern are best treated by some pre-computed rules. What is interesting is the retrieval of lexicalized rules. More precisely, we want to retrieve efficiently treelets containing at least one leaf of the query tree. Therefore, an alternative computation strategy would only explore treelets containing at least one terminal node. We would thus compute successively "dog", "NN(dog)" "NP(NN(dog))", "NP(Det NN(dog))", etc.

4.4 Complexity

Processing time will be mainly dependent on two factors: the number of treelets in a query tree that need to be computed, and the average time to compute a treelet.

Let N_C be the size of the corpus. It can be shown quite easily that the time needed to compute a treelet with our method is proportional to its number of occurrences, which is itself growing as $O(N_C)$.

Let m be the size of the query tree. The number of treelets needing to be computed is, in the worst case, exponential in m. In practice, the only case where most of the treelets are non-empty is when the database contains trees similar to the query tree in the database, and this is handled by the modification of the algorithm is section 4.2. In other cases, most of the treelets are empty, and empirically, we find that the number of non-empty treelets in a query tree

Database size (#nodes)	6M	60M
Largest non-empty treelet size	4.6	8.7
Processing time (PtR Array)	0.02 s	0.7 s
Processing time (Inv. Index)	0.02 s	0.9 s
Size on disk	40 MB	500 MB

Figure 5: Performances averaged on 100 sentences.

Method	Treelet dictionary	Our method
Disk space used	23 GB	500 MB
BLEU	11.6%	12.0%

Figure 6: Comparison with a dictionary-based baseline (performances averaged over 100 sentences).

grows approximately as $O(m \cdot N_C^{0.5})$. It is also possible to bound the size of the retrieved treelets (only retrieving treelets with less than 10 nodes, for example), similarly to what is done in (Lopez, 2007). The number of treelets will then only grows as $O(m)$.

The total processing time of a given query tree will therefore be on the order of $O(m \cdot N_C^{1.5})$ (or $O(m \cdot N_C)$ if we bound the treelet size). The fact that this give a complexity worse than linear with respect to the database size might seem a concern, but this is actually only because we are retrieving more and more different types of treelets. The cost of retrieving one treelet remain linear with respect to the size of the corpus. We empirically find that even for very large values of N_C, processing time remain very reasonable (see next section).

It should be also noted that the constant hidden in the big-O notation can be (almost) arbitrarily reduced by precomputing more and more of the most common (and more expensive) treelets (a time-memory trade-off).

5 Experiments

We conducted experiments on a large database of 2.9 million automatically parsed dependency trees, with a total of nearly 60 million nodes[1]. The largest trees in the database have around 100 nodes. In order to see how performance scale with the size of the database, we also used a smaller subset of 230,000 trees containing near 6 million nodes.

We computed, using our algorithm, 100 randomly selected query trees having from 10 to 70 nodes, with an average of 27 nodes per tree. Table 5 shows the average performances per sentence. Considering the huge size of the database, a processing time below 1 second seems reasonable. The increase in processing time between the small and the large database is in line with the explanations of section 4.4. Path-to-Root Arrays are slightly better than Inverted indexes (we suspect a better implementation could increase the difference further). Both methods use up about the same disk space: around 500MB. We also find that the approach of section 4.2 brings virtually no overhead and gives similar performances whether the query tree is in the database or not (effectively reducing the worst-case computation time from days to seconds).

We also conducted a small English-to-Japanese translation experiment with a simple translation system using Synchronous Tree Substitution Grammars (STSG) for translating dependency trees. The system we used is still in an experimental state and probably not quite at the state-of-the-art level yet. However, we considered it was good enough for our purpose, since we mainly want to test our algorithm is a practical way. As a baseline, from our corpus of 2.9 millions dependency trees, we automatically extracted STSG rules of size smaller than 6 and stored them in a database, considering that extracting rules of larger sizes would lead to an unmanageable database size. We compared MT results using only the rules of size smaller than 6 to using all the rules computed on-the-fly after treelet retrieving by our method. These results are summarized on figure 6.

6 Flexible matching

We now describe an extension of the algorithm for approximate matching of treelets. We consider that each node of the query tree and database is labeled by 2 labels (or more) of different generality. For concreteness, let's consider dependency trees whose nodes are labeled by words and the Part-Of-Speech (POS) of these words. We want to retrieve treelets

[1]This database was an aggregate of several Japanese-English corpora, notably the Yomiuri newspaper corpus (Utiyama and Isahara, 2003) and the JST paper abstract corpus created at NICT(www.nict.go.jp) through (Utiyama and Isahara, 2007).

that match by word or POS with the query tree.

6.1 Processing multi-Label trees

To do this, the inverted index will just need to include entries for both words and POS. For example, the dependency tree "likes,V:1 (Paul,N:0 Salmon,N:2 (and,CC:3 (Tuna,N:4)))" would produce the following *(node,parents)* entries in the inverted index: {N:[(0,1) (2,1) (4,3)], Paul:[(0,1)], Salmon:[(2,1)],...}. This allows to search for a treelet containing any combination of labels, like "likes(N Salmon(CC(N)))".

We actually want to compute all of the treelets of a query tree $\mathcal{T}_Q$ labeled by words and POS (meaning each node can be matched by either word or POS).

We can compute $\mathcal{T}_Q$ without redundant computations by slightly modifying the algorithm 2. First, we modify the recursive representation of a treelet so that it also includes the chosen label of its root node. Then, the only modifications needed in algorithm 2 are the following: 1- at initialization (line 3), the elementary treelets corresponding to every possible labels are added to the discovered treelets set $\mathcal{D}$; 2- in procedure *compute-supertrees*, at line 8, we generate one ascending supertreelet per label.

6.2 Weighted search

While the previous method would allow us to compute as efficiently as possible all the treelets included in a multi-labeled query tree, there is still a problem: even avoiding redundant computations, the number of treelets to compute can be huge, since we are computing all combinations of labels. For each treelet of size m we would have had in a single label query tree, we now virtually have 2^m treelets. Therefore, it is not reasonable in general to try to compute all these treelets.

However, we are not really interested in computing all possible treelets. In our case, the POS labels allow us to retrieve larger examples when none containing only words would be available. But we still prefer to find examples matched by words rather than by POS. We therefore need to tell the algorithm that some treelets are more important that some others. While we have used the Computation Hypertree representation to compute treelets efficiently, we can also use it to prioritize the treelets we want to compute. This is easily implemented by giving a weight to every treelet. We can then modify our traversal strategy of the Inclusion DAG to compute treelets having the biggest weights first: we just need to specify that the treelet popped out on line 3 is the treelet with the highest score (more generally, we could consider a A* search).

POS matchings	Without	With
Processing time	0.9 s	22 s
Largest non-empty treelet size	8.7	11.4
Treelets of size>8	0.4	102
BLEU	12.0%	12.1%

Figure 7: Effect of POS-matching

6.3 Experiments

Using the above ideas, we have made some experiments for computing query dependency trees labeled with both words and POS. We score the treelets by giving them a penalty of -1 for each POS they contain, and stop the search when all remaining treelets have a score lower than -2 (in other words, treelets are allowed at most 2 POS-matchings). We also require POS-matched nodes to be non-adjacent.

We only have some small modifications to do to algorithm 2. In line 3 of algorithm 2, elementary treelets are assigned a weight of 0 or -1 depending on whether their label is a word or POS. Line 5 is replaced by "pop the first treelet with minimal weight and break the loop if the minimal weight is inferior to -2". In *compute-supertreelets*, we give a weight to the generated supertreelets by combining the weights of the child treelets.

Table 7 shows the increase in the size of the biggest non-empty treelets when allowing 2 nodes to be matched by POS. It also shows the impact on BLEU score of using these additional treelets for on-the-fly rule generation in our simple MT system. Improvement on BLEU is limited, but it might be due to a very experimental handling of approximately matched treelet examples in our MT system.

The computation time, while manageable, was much slower than in the one-label case. This is due to the increased number of treelets to be computed, and to the fact that POS-labeled elementary treelets have a high number of occurrences. It would be more efficient to use more specific labeling (e.g V-

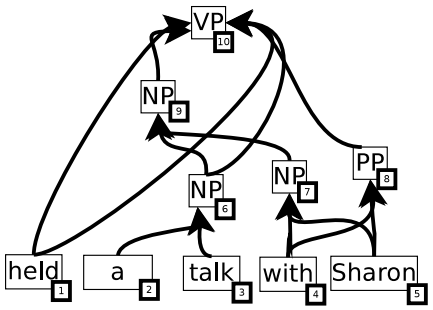


Figure 8: A packed forest.

mvt for verbs of movement instead of V).

7 Additional extensions

We briefly discuss here some additional extensions to our algorithm that we will not detail for lack of room and practical experiments.

7.1 Packed forest

Due to parsing ambiguities and automatic parsers errors, it is often useful to use multiple parses of a given sentence. These parses can be represented by a packed forest such as the one in figure 8. Our method allows the use of packed representation of both the query tree and the database.

For the inverted index, the only difference is that now, an occurrence of a label can have more than one parent. For example, the inverted index of a database containing the packed forest of figure 8 would contain the following entries: {held: [(1,10a),(1,10b)], NP: [(6,9),(7,9),(9,10a)], VP:[(10,N)], PP:[(8,10b)], a:[(2,6)], talk:[(3,6)], with:[(4,7) (4,8)], Sharon:[(5,7) (5,8)]}. Where 10a and 10b are some kind of virtual position that help to specify that *held* and NP_8 belong to the same children list. We could also include a cost on edges in the inverted index, which would allow to prune matchings to unlikely parses.

The inverted index can now be used to search in the trees contained in a packed forest database without any modification. Modifications to the algorithm in order to handle a packed forest query are similar to the ones developed in section 6.

7.2 Discontinuous treelets

As we discussed in section 2.3, using a representation for the position of every node similar to (Zhang et al., 2001), it is possible to determine the distance and ancestor relationship of two nodes by just comparing their positions. This opens the possibility of computing the occurrences of discontinuous treelets in much the same way as is done in (Lopez, 2007) for discontinuous substrings. We have not studied this aspect in depth yet, especially since we are not aware of any MT system making use of discontinuous syntax tree examples. This is nevertheless an interesting future possibility.

8 Related work

As we previously mentioned, (Lopez, 2007) and (Callison-Burch et al., 2005) propose a method similar to ours for the string case.

We are not aware of previous proposals for efficient on-the-fly retrieving of translation examples in the case of Syntax-Based Machine Translation. Among the works involving rule precomputation, (Zhang et al., 2009) describes a method for efficiently matching precomputed treelets rules. These rules are organized in a kind of prefix tree that allows efficient matching of packed forests. (Liu et al., 2006) also propose a greedy algorithm for matching TSC rules to a query tree.

9 Conclusion and future work

We have presented a method for efficiently retrieving examples of treelets contained in a query tree, thus allowing on-the-fly computation of translation rules for Syntax-Based systems. We did this by building on approaches previously proposed for the case of string examples, proposing an adaptation of the concept of Suffix Arrays to trees, and formalizing computation as the traversal of an hypergraph. This hypergraph allows us to easily formalize different computation strategy, and adapt the methods to flexible matchings. We still have a lot to do with respect to improving our implementation, exploring the different possibilities offered by this framework and proceeding to more experiments.

Acknowledgments

We thank the anonymous reviewers for their useful comments.

References

R. Baeza-Yates and A. Salinger. 2005. Experimental analysis of a fast intersection algorithm for sorted sequences. In *String Processing and Information Retrieval*, page 1324.

N. Bruno, N. Koudas, and D. Srivastava. 2002. Holistic twig joins: optimal XML pattern matching. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, page 310321.

C. Callison-Burch, C. Bannard, and J. Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 255–262. Association for Computational Linguistics Morristown, NJ, USA.

David Chiang. 2007. Hierarchical Phrase-Based translation. *Computational Linguistics*, 33(2):201–228, June.

M. Dubiner, Z. Galil, and E. Magen. 1994. Faster tree pattern matching. *Journal of the ACM (JACM)*, 41(2):205213.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 48–54. Association for Computational Linguistics.

Z. Liu, H. Wang, and H. Wu. 2006. Example-based machine translation based on tree–string correspondence and statistical generation. *Machine translation*, 20(1):25–41.

A. Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proc. of EMNLP-CoNLL*, page 976985.

U. Manber and G. Myers. 1990. Suffix arrays: a new method for on-line string searches. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 319–327, San Francisco, CA, USA. Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

H. Mi, L. Huang, and Q. Liu. 2008. Forest based translation. *Proceedings of ACL-08: HLT*, page 192199.

Toshiaki Nakazawa and Sadao Kurohashi. 2008. Syntactical EBMT system for NTCIR-7 patent translation task. In *Proceedings of NTCIR-7 Workshop Meeting*, Tokyo, Japon.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, page 279.

M. Utiyama and H. Isahara. 2003. Reliable measures for aligning japanese-english news articles and sentences. In *Proceedings of ACL*, pages 72–79, Sapporo,Japon.

M. Utiyama and H. Isahara. 2007. A japanese-english patent parallel corpus. In *MT summit XI*, pages 475–482.

M. Yamamoto and K. W. Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1):1–30.

Y. Zhang and S. Vogel. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of EAMT*, pages 294–301, Budapest, Hungary.

C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. 2001. On supporting containment queries in relational database management systems. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, page 425436.

H. Zhang, M. Zhang, H. Li, and Chew Lim Tan. 2009. Fast translation rule matching for syntax-based statistical machine translation. In *Proc. of EMNLP*, pages 1037–1045.

A generative model for unsupervised discovery of relations and argument classes from clinical texts

Bryan Rink and Sanda Harabagiu
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX, USA
{bryan,sanda}@hlt.utdallas.edu

Abstract

This paper presents a generative model for the automatic discovery of relations between entities in electronic medical records. The model discovers relation instances and their types by determining which context tokens express the relation. Additionally, the valid semantic classes for each type of relation are determined. We show that the model produces clusters of relation trigger words which better correspond with manually annotated relations than several existing clustering techniques. The discovered relations reveal some of the implicit semantic structure present in patient records.

1 Introduction

Semantic relations in electronic medical records (EMRs) capture important meaning about the associations between medical concepts. Knowledge about how concepts such as medical problems, treatments, and tests are related can be used to improve medical care by speeding up the retrieval of relevant patient information or alerting doctors to critical information that may have been overlooked. When doctors write progress notes and discharge summaries they include information about how treatments (e.g., aspirin, stent) were administered for problems (e.g. pain, lesion) along with the outcome, such as an improvement or deterioration. Additionally, a doctor will describe the tests (e.g., x-ray, blood sugar level) performed on a patient and whether the tests were conducted to investigate a known problem or revealed a new one. These textual descriptions written in a patient's record encode important information about the relationships between the problems a patients has, the treatments taken for the problems, and the tests which reveal and investigate the problems.

The ability to accurately detect semantic relations in EMRs, such as *Treatment-Administered-for-Problem*, can aid in querying medical records. After a preprocessing phase in which the relations are detected in all records they can be indexed and retrieved later as needed. A doctor could search for all the times that a certain treatment has been used on a particular problem, or determine all the treatments used for a specific problem. An additional application is the use of the relational information to flag situations that merit further review. If a patient's medical record indicates a test that was found to reveal a critical problem but no subsequent treatment was performed for the problem, the patient's record could be flagged for review. Similarly, if a *Treatment-Worsens-Problem* relation is detected previously in a patient's record, that information can be brought to the attention of a doctor who advises such a treatment in the future. By considering all of the relations present in a corpus, better medical ontologies could be built automatically or existing ones can be improved by adding additional connections between concepts that have a relation in text.

Given the large size of EMR repositories, we argue that it is quite important to have the ability to perform relation discovery between medical concepts. Relations between medical concepts benefit translational medicine whenever possible relations are known. Uzuner et al. (2011) show that super-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 519–528,
Edinburgh, Scotland, UK, July 27–31, 2011.

vised methods recognize such relations with high accuracy. However, large sets of annotated relations need to be provided for this purpose. To address both the problem of discovering unknown relations between medical concepts and the related problem of generating examples for known relations, we have developed an unsupervised method. This approach has the advantages of not requiring an expensive annotation effort to provide training data for semantic relations, which is particularly difficult for medical records, characterized by many privacy concerns. Our analysis shows a high level of overlap between the manually annotated relations and those that were discovered automatically. Our experimental results show that this approach improves upon simpler clustering techniques.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 reports our novel generative model for discovering relations in EMRs, Section 4 details the inference and parameter estimation of our method. Section 5 details our experiments, Section 6 discusses our findings. Section 7 summarizes the conclusions.

2 Related Work

Previous methods for unsupervised relation discovery have also relied on clustering techniques. One technique uses the context of entity arguments to cluster, while another is to perform a post-processing step to cluster relations found using an existing relation extraction system. The approaches most similar to ours have taken features from the context of pairs of entities and used those features to form a clustering space. In Hasegawa et al. (2004), those features are tokens found within a context window of the entity pair. Distance between entity pairs is then computed using cosine similarity. In another approach, Rosenfeld and Feldman (2007) use hierarchical agglomerative clustering along with features based on token patterns seen in the context, again compared by cosine similarity.

Other approaches to unsupervised relation discovery have relied on a two-step process where a number of relations are extracted, usually from a predicate-argument structure. Then similar relations are clustered together since synonymous predicates should be considered the same relation (e.g. "acquire" and "purchase"). Yates (2009) considers the output from an open information extraction system (Yates et al., 2007) and clusters predicates and arguments using string similarity and a combination of constraints. Syed and Viegas (2010) also perform a clustering on the output of an existing relation extraction system by considering the number of times two relations share the same exact arguments. Similar relations are expected to have the same pairs of arguments (e.g. "Ford produces cars" and "Ford manufactures cars"). These approaches and others (Agichtein and Gravano, 2000; Pantel and Pennacchiotti, 2006) rely on an assumption that relations are context-independent, such as when a person is born, or the capital of a nation. Our method will discover relations that can depend on the context as well. For instance, "penicillin" may be causally related to "allergic reaction" in one patient's medical record but not in another. The relation between the two entities is not globally constant and should be considered only within the scope of one patient's records.

Additionally, these two-step approaches tend to rely on predicate-argument structures such as subject-verb-object triples to detect arbitrary relations (Syed and Viegas, 2010; Yates et al., 2007). Such approaches can take advantage of the large body of research that has been done on extracting syntactic parse structure and semantic role information from text. However, these approaches can overlook relations in text which do not map easily onto those structures. Unlike these approaches, our model can detect relations that are not expressed as a verb, such as "[cough] + [green sputum]" to express a conjunction or "[Cl] 119 mEq / L [High]" to express that a test reading is indicating a problem.

The 2010 i2b2/VA Challenge (Uzuner et al., 2011) developed a set of annotations for medical concepts and relations on medical progress notes and discharge summaries. One task at the challenge involved developing systems for the extraction of eight types of relations between concepts. We use this data set to compare our unsupervised method with others.

The advantage of our work over existing unsupervised approaches is the simultaneous clustering of both argument words and relation trigger words. These broad clusters handle: (i) synonyms, (ii) argu-

ment semantic classes, and (iii) words belonging to the same relation.

3 A Generative Model for Discovering Relations

3.1 Unsupervised Relation Discovery

A simple approach to discovering relations between medical entities in clinical texts uses a clustering approach, e.g. Latent Dirichlet Allocation (LDA) (Blei et al., 2003). We start with an assumption that relations exist between two entities, which we call arguments, and may be triggered by certain words between those entities which we call *trigger words*. For example, given the text "[x-ray] revealed [lung cancer]", the first argument is *x-ray*, the second argument is *lung cancer*, and the trigger word is *revealed*. We further assume that the arguments must belong to a small set of semantic classes specific to the relation. For instance, *x-ray* belongs to a class of medical tests, whereas *lung cancer* belongs to a class of medical problems. While relations may exist between distant entities in text, we focus on those pairs of entities in text which have no other entities between them. This increases the likelihood of a relation existing between the entities and minimizes the number of context words (words between the entities) that are not relevant to the relation.

With these assumptions we build a baseline relation discovery using LDA. LDA is used as a baseline because of its similarities with our own generative model presented in the next section. Each consecutive pair of entities in text is extracted, along with the tokens found between them. Each of the entities in a pair is split into tokens which are taken along with the context tokens to form a single *pseudo-document*. When the LDA is processed on all such pseudo-documents, clusters containing words which co-occur are formed. Our assumption that relation arguments come from a small set of semantic classes should lead to clusters which align with relations since the two arguments of a relation will co-occur in the pseudo-documents. Furthermore, those argument tokens should co-occur with relation trigger words as well.

This LDA-based approach was examined on electronic medical records from the 2010 i2b2/VA Challenge data set (Uzuner et al., 2011). The data set contains manually annotated medical entities which were used to form the pairs of entities needed. For example, Figure 1 illustrates examples of two clusters out of 15 discovered automatically using LDA on the corpus. The first cluster appears to contain words which indicate a relation whose two arguments are both medical problems (e.g. "disease", "cancer", "edema"). The trigger words seem to indicate a possible causal relation (e.g., "due", "related", "secondary"). The second cluster contains words relevant to medical tests (e.g. "examination", "culture") and their findings ("revealed", "showed", "positive"). As illustrated in Figure 1, some of the context words are not necessarily related to the relation. The word "patient" for instance is present in both clusters but is not a trigger word because it is likely to be seen in the context of any relation in medical text. The LDA-based model treats all words equally and cannot identify which words are likely trigger words and which ones are *general words*, which merely occur frequently in the context

Cluster 1
Words: secondary, due, likely, patient, disease, liver, abdominal, cancer, pulmonary, respiratory, elevated, volume, chronic, edema, related
"Correct" instances: [Metastatic colon cancer] with [abdominal carcinomatosis]; [symptoms] were due to [trauma]
"Incorrect" instances: [mildly improving symptoms] , plan will be to continue with [his current medicines]; [prophylaxis] against [peptic ulcer disease]
Cluster 2:
Words: examination, no, positive, culture, exam, blood, patient, revealed, cultures, physical, out, urine, notable, showed, cells
"Correct" instances: [a blood culture] grew out [Staphylococcusaureus]; [tamponade] by [examination]
"Incorrect" instances: [the intact drain] draining [bilious material]; [a Pseudomonas cellulitis] and [subsequent sepsis]

Figure 1: Two clusters found by examining the most likely words under two LDA topics. The instances are pseudo-documents whose probability of being assigned to that cluster was over 70%

of a relation.

In addition, while the LDA approach can detect argument words which co-occur with trigger words (e.g., "examination" and "showed"), the clusters produced with LDA do not differentiate between contextual words and words which belong to the arguments of the relation. An approach which models arguments separately from context words could learn the semantic classes of those arguments and thus better model relations. Considering the examples from Figure 1, a model which could cluster "examination", "exam", "cultures", and "culture" into one *medical test* cluster and "disease", "cancer" and "edema" into a *medical problem* cluster separate from the relation trigger words and general words should model relations more accurately by better reflecting the implicit structure of the text. Because of these limitations many relations discovered in this way are not accurate, as can be seen in Figure 1.

3.2 Relation Discovery Model (RDM)

The limitations identified in the LDA-based approach are solved by a novel relation discovery model (RDM) which jointly models relation argument semantic classes and considers them separately from the context words. Relations triggered by pairs of medical entities enable us to consider three observable features: (A1) the first argument; (A2) the second argument; and (CW) the context words found between A1 and A2.

For instance, in sentence S1 the arguments are A1="some air hunger" and A2="his tidal volume" while the context words are "last", "night", "when", "I", and "dropped".

S1: *He developed [some air hunger]$_{PROB}$ last night when I dropped [his tidal volume]$_{TREAT}$ from 450 to 350.*

In the RDM, the contextual words are assumed to come from a mixture model with 2 mixture components: a relation trigger word ($x = 0$), or a general word ($x = 1$), where x is a variable representing which mixture component a word belongs to. In sentence S1 for example, the word "dropped" can be seen as a trigger word for a *Treatment-Causes-Problem* relation. The remaining words are not trigger words and hence are seen as general words.

Under the RDM's mixture model, the probability of a context word is:

$$P(w^C|t^r, z) = P(w^C|t^r, x = 0) \times P(x = 0|t^r) + P(w^C|z, x = 1) \times P(x = 1|t^r)$$

Where w^C is a context word, the variable t^r is the relation type, and z is the general word class. The variable x chooses whether a context word comes from a relation-specific distribution of trigger words, or from a general word class. In the RDM, the two argument classes are modeled jointly as $P(c^1, c^2|t^r)$, where c^1 and c^2 are two semantic classes associated with a relation of type t^r. However the assignment of classes to arguments depends on a directionality variable d. If $d = 0$, then the first argument is assigned semantic class c^1 and the second is assigned class c^2. When $d = 1$ however, the class assignments are swapped. This models the fact that a relation's arguments do not come in a fixed order, "[MRI] revealed [tumor]" is the same type of relation as "[tumor] was revealed by [x-ray]". Figure 2 shows the graphical model for the RDM. Each candidate relation is modeled independently, with a total of I relation candidates. Variable w^1 is a word observed from the first argument, and w^2 is a word observed from the second argument. The model takes parameters for the number of relations types (R), the number of argument semantic classes (A), and the number of general word classes (K). The generative process for the RDM is:

1. For relation type $r = 1..R$:
   (a) Draw a binomial distribution σ_r from $Beta(\alpha^x)$ representing the mixture distribution for relation r
   (b) Draw a joint semantic class distribution $\psi_r^{1,2} \in \mathbb{R}^{C \times C}$ from $Dirichlet(\alpha^{1,2})$.
2. Draw a categorical word distribution $\phi_{z'}^z$ from $Dirichlet(\beta^z)$ for each general word class $z' = 1..K$
3. Draw a categorical word distribution $\phi_{r'}^r$ from $Dirichlet(\beta^r)$ for each $r' = 1..R$
4. for semantic class $a' = 1..A$:
   (a) Draw categorical word distributions $\omega_{a'}^1$ and $\omega_{a'}^2$ from $Dirichlet(\beta^1)$ and $Dirichlet(\beta^2)$ for the first and second arguments, respectively.

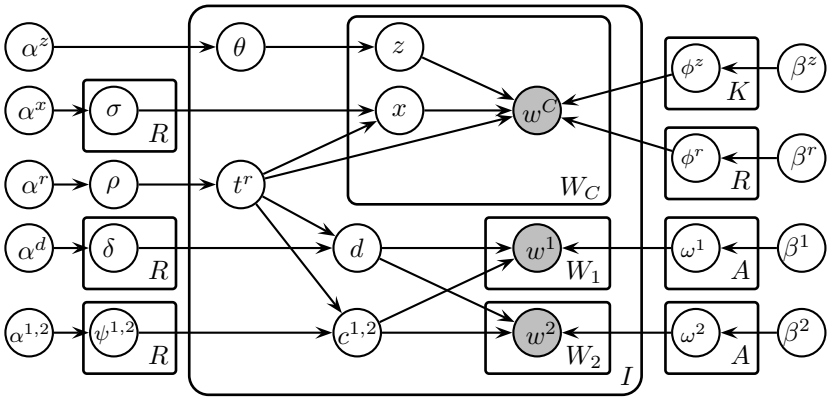


Figure 2: Graphical model for the RDM. $c^{1,2}$ represents the joint generation of c^1 and c^2

$$P(t^r, d|\boldsymbol{t^r_{-i}}, \boldsymbol{d_{-i}}, \boldsymbol{c^{1,2}_{-i}}, \boldsymbol{x_{-i}}, \boldsymbol{z_{-i}}, \boldsymbol{w^C_{-i}}, \boldsymbol{w^1_{-i}}, \boldsymbol{w^2_{-i}}; \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto u_1 \times u_2 \times u_3$$

$$u_1 = \frac{f(t^r)+\alpha^r}{I+R\alpha^r} \times \frac{f(t^r,d)+\alpha^d_d}{f(t^r)+\alpha^d_0+\alpha^d_1} \times \frac{f(t^r,c^1,c^2)+\alpha^{1,2}}{f(t^r)+C\times C\alpha^{1,2}}$$

$$u_2 = \prod_j^{W_C} \frac{f_i(z_j)+\alpha^z}{W_C+K\alpha^z} \times \frac{f(t^r,x_i)+\alpha^x}{f(t^r)+2\alpha^x} \times (\mathbf{1}_{x=0}\frac{f(t^r,w^C_j)+\beta^r}{f(t^r)+W\beta^r} + \mathbf{1}_{x=1}\frac{f(z_j,w^C_j)+\beta^z}{f(z_j)+W\beta^z})$$

$$u_3 = \prod_j^{W_1} \frac{f(a^1,w^1_j)+\beta^1}{f(a^1)+W\beta^1} \times \prod_j^{W_2} \frac{f(a^2,w^2_j)+\beta^2}{f(a^2)+W\beta^2}$$

Figure 3: Gibbs sampling update equation for variables t^r and d for the i^{th} relation candidate. The variables $a^1 = c^1$ and $a^2 = c^2$ if $d = 0$, or $a^1 = c^2$ and $a^2 = c^1$ if $d = 1$. W is the size of the vocabulary. $f(\bullet)$ is the count of the number of times that event occurred, excluding assignments for the relation instance being sampled. For instance, $f(t^r, d) = \sum_{k\neq i}^{I} I[t^r_k = t^r_i \wedge d_k = d_i]$

5. Draw a categorical relation type distribution ρ from $Dirichlet(\alpha^r)$
6. For each pair of consecutive entities in the corpus, $i = 1..I$:
   (a) Sample a relation type t^r from ρ
   (b) Jointly sample semantic classes c^1 and c^2 for the first and second arguments from $\psi^{1,2}_{t^r}$
   (c) Draw a general word class categorical distribution θ from $Dirichlet(\alpha^z)$
   (d) For each token $j = 1..W_1$ in the first argument: Sample a word w^1_j from $\omega^1_{c^1}$ if $d = 0$ or $\omega^2_{c^2}$ if $d = 1$
   (e) For each token $j = 1..W_2$ in the second argument: Sample a word w^2_j from $\omega^2_{c^2}$ if $d = 0$ or $\omega^1_{c^1}$ if $d = 1$
   (f) For each token $j = 1..W_C$ in the context of the entities:
      i. Sample a general word class z from θ
      ii. Sample a mixture component x from σ_{t^r}
      iii. Sample a word from $\phi^r_{t^r}$ if $x = 0$ or ϕ^z_z if $x = 1$.

In the RDM, words from the arguments are informed by the relation through an argument semantic class which is sampled from $P(c^1, c^2|t^r) = \psi^{1,2}_{t^r}$. Furthermore, words from the context are informed by the relation type. These dependencies enable more coherent relation clusters to form during parameter estimation because argument classes and relation trigger words are co-clustered.

We chose to model two distinct sets of entity words (ω^1 and ω^2) depending on whether the entity occurred in the first argument or the second argument of the relation. The intuition for using disjoint sets of entities is based on the observation that an entity may be expressed differently if it comes first or second in the text.

4 Inference and Parameter Estimation

Assignments to the hidden variables in RDM can be made by performing collapsed Gibbs sampling (Griffiths and Steyvers, 2004). The joint probability of the data is:

$$P(\boldsymbol{w^C}, \boldsymbol{w^1}, \boldsymbol{w^2}; \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto$$
$$P(\sigma|\alpha^x)P(\rho|\alpha^r)P(\delta|\alpha^d)P(\psi^{1,2}|\alpha^{1,2})$$
$$\times P(\phi^z|\beta^z)P(\phi^r|\beta^r)P(\omega^1|\beta^1)P(\omega^2|\beta^2)$$
$$\times \prod_i^I [P(\theta_i|\alpha^z)P(t_i^r|\rho)P(d_i|t^r, \delta_{t^r})P(c_i^1, c_i^2|t^r, \psi^{1,2})$$
$$\times \prod_j^{W_{C,i}} P(z_{i,j}|\theta_i)P(x_{i,j}|t_i^r, \sigma_{t_i^r})P(w_{i,j}^C|x_{i,j}, t_i^r, z_{i,j})$$
$$\times \prod_j^{W_{1,i}} P(w_j^1|d_i, c_i^{1,2}, \omega^1)$$
$$\times \prod_j^{W_{2,i}} P(w_j^2|d_i, c_i^{1,2}, \omega^2)]$$

We need to sample variables t^r, d, $c^{1,2}$, x, and z. We sample t^r and d jointly while each of the other variables is sampled individually. After integrating out the multinomial distributions, we can sample t^r and d from the equation in Figure 3

The update equations for the remaining variables can be derived from the same equation by dropping terms which are constant across changes in that variable.

In our experiments the hyperparameters were set to $\alpha^x = 1.0, \alpha^z = 1.0, \alpha^{1,2} = 1.0, \alpha_0^d = 2, \alpha_1^d = 1, \beta^r = 0.01, \beta^z = 0.01, \beta^1 = 1.0, \beta^2 = 1.0$. Changing the hyperparameters did not significantly affect the results.

5 Experimental Results

5.1 Experimental Setup

We evaluated the RDM using a corpus of electronic medical records provided by the 2010 i2b2/VA Challenge (Uzuner et al., 2011). We used the training set, which consists of 349 medical records from 4 hospitals, annotated with medical concepts (specifically problems, treatments, and tests), along with any relations present between those concepts. We used these manually annotated relations to evaluate how well the RDM performs at relation discovery. The corpus is annotated with a set of eight relations: *Treatment-Addresses-Problem*, *Treatment-Causes-Problem*, *Treatment-Improves-Problem*, *Treatment-Worsens-Problem*, *Treatment-Not-Administered-due-to-Problem*, *Test-Reveals-Problem*, *Test-Conducted-for-Problem*, and *Problem-Indicates-Problem*. The data contains 13,460 pairs of consecutive concepts, of which 3,613 (26.8%) have a relation belonging to the list above. We assess the model using two versions of this data set consisting of: those pairs of consecutive entities which have a manually annotated relation (DS1), and secondly, all consecutive pairs of entities (DS2). DS1 allows us to assess the RDM's clustering without the noise introduced from those pairs lacking a true relation. Evaluations on DS2 will indicate the level of degradation caused by large numbers of entity pairs that have no true relation. We also use a separate test set to assess how well the model generalizes to new data. The test set contains 477 documents comprising 9,069 manually annotated relations.

Relation 1	Relation 2	Relation 3	Relation 4
mg	(	due	showed
p.r.n.	)	consistent	no
p.o.	Working	not	revealed
hours	ICD9	likely	evidence
prn	Problem	secondary	done
q	Diagnosis	patient	2007
needed	30	(	performed
day	cont	started	demonstrated
q.	):	most	without
4	closed	s/p	normal
2	SNMCT	seen	shows
every	**ID-NUM	related	found
one	PRN	requiring	showing
two	mL	including	negative
8	ML	felt	well

Figure 4: Relation trigger words found by the RDM

5.2 Analysis

Figure 4 illustrates four of the fifteen trigger word clusters (most likely words according to ϕ^r) learned from dataset DS1 using the best set of parameters according to normalized mutual information (NMI) as described in section 5.3. These parameters were: $R = 9$ relations, $K = 15$ general word classes, and $A = 15$ argument classes. Examination of the most likely words reveals a variety of trigger words, beyond obvious explicit ones. Example sentences for the relation types from Figure 4 are presented in Figure 5 and discussed below.

Relation Type 1

Instances of this discovered relation are often found embedded in long lists of drugs prescribed to the patient. Tokens such as "p.o." and "p.r.n.", meaning respectively "by mouth" and "when necessary", are indicative of a prescription relation. The learned relation specifically considers arguments of a drug

Instances of Relation Type 1

1. Haldol 0.5-1 milligrams p.o. q.6-8h. p.r.n. agitation
2. plavix every day to prevent failure of these stents
3. KBL mouthwash , 15 ccp .o. q.d. prn mouth discomfort
4. Miconazole nitrate powder tid prn for groin rash
5. AmBisome 300 mg IV q.d. for treatment of her hepatic candidiasis

Instances of Relation Type 2

1. MAGNESIUM HYDROXIDE SUSP 30 ML) , 30 mL , Susp , By Mouth , At Bedtime , PRN , For Constipation
2. Depression , major (ICD9 296.00 , Working , Problem) cont NOS home meds
3. Diabetes mellitus type II (ICD9 250.00 , Working , Problem) cont home meds
4. ASCITES (ICD9 789.5 , Working , Diagnosis) on spironalactone
5. *Dilutional hyponatremia (SNMCT **ID-NUM , Working , Diagnosis) improved with fluid restriction

Instances of Relation Type 3

1. ESRD secondary to her DM
2. slightly lightheaded and with increased HR
3. a 40% RCA , which was hazy
4. echogenic kidneys consistent with renal parenchymal disease
5. *Librium for alcohol withdrawal

Instances of Relation Type 4

1. V-P lung scan was performed on May 24 2007 , showed low probability of PE
2. a bedside transthoracic echocardiogram done in the Cardiac Catheterization laboratory without evidence of an effusion
3. exploration of the abdomen revealed significant nodularity of the liver
4. Echocardiogram showed moderate dilated left atrium
5. An MRI of the right leg was done which was equivocal for osteomyelitis

Figure 5: Examples for four of the discovered relations. Those marked with an asterisk have a different manually chosen relation than the others

and a symptom treated by that drug. The closest manually chosen relation is *Treatment-Addresses-Problem* which included drugs as treatments.

Relation Type 2

Relation 2 captures a similar kind of relation to Relation 1. All five examples for Relation 1 in Figure 5 came from a different set of hospitals than the examples for Relation 2. This indicates the model is detecting stylistic differences in addition to semantic differences. This is one of shortcomings of simple generative models. Because they cannot reflect the true underlying distribution of the data they will model the observations in ways that are irrelevant to the task at hand. Relation 2 also contains certain punctuation, such as parentheses which the examples show are used to delineate a treatment code. Instances of Relation 2 were often marked as *Treatment-Addresses-Problem* relations by annotators.

Relation Type 3

The third relation captures problems which are related to each other. The manual annotations contain a very similar relation called *Problem-Indicates-Problem*. This relation is also similar to Cluster 1 from Section 3.1, however under the RDM the words are much more specific to the relation. This relation is difficult to discover accurately because of the infrequent use of strong trigger words to indicate the relation. Instead, the model must rely more on the semantic classes of the arguments, which in this case will both be types of medical problems.

Relation Type 4

The fourth relation is detecting instances where a medical test has revealed some problem. This corresponds to the *Test-Reveals-Problem* relation from the data. Many good trigger words for that relation have high probability under Relation 4. A comparison of the RDM's Relation 4 with LDA's cluster 2 from Figure 1 shows that many words not relevant to the relation itself are now absent.

Argument classes

Figure 6 shows the 3 most frequent semantic classes

Concept 1	Concept 2	Concept 3
CT	pain	Percocet
scan	disease	Hgb
chest	right	Hct
x-ray	left	Anion
examination	renal	Vicodin
Chest	patient	RDW
EKG	artery	Bili
MRI	-	RBC
culture	symptoms	Ca
head	mild	Gap

Figure 6: Concept words found by the RDM

for the first argument of a relation (ω^1). Most of the other classes were assigned rarely, accounting for only 19% of the instances collectively. Human annotators of the data set chose three argument classes: *Problems*, *Treatments*, and *Tests*. Concept 1 aligns closely with a test semantic class. Concept 2 seems to be capturing medical problems and their descriptions. Finally, Concept 3 appears to be a combination of treatments (drugs) and tests. Tokens such as "Hgb", "Hct", "Anion", and "RDW" occur almost exclusively in entities marked as tests by annotators. It is not clear why this cluster contains both types of words, but many of the high ranking words beyond the top ten do correspond to treatments, such as "Morphine", "Albumin", "Ativan", and "Tylenol". Thus the discovered argument classes show some similarity to the ones chosen by annotators.

5.3 Evaluation

For a more objective analysis of the relations detected, we evaluated the discovered relation types by comparing them with the manually annotated ones from the data using normalized mutual information (NMI) (Manning et al., 2008). NMI is an information-theoretic measure of the quality of a clustering which indicates how much information about the gold classes is obtained by knowing the clustering. It is normalized to have a range from 0.0 to 1.0. It is defined as:

$$NMI(\Omega;\mathbb{C}) = \frac{I(\Omega;\mathbb{C})}{[H(\Omega)+H(\mathbb{C})]/2}$$

where Ω is the system-produced clustering, $\mathbb{C}$ is the gold clustering, I is the mutual information, and H is the entropy. The mutual information of two clusterings can be defined as:

$$I(\Omega,\mathbb{C}) = \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} log_2 \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|}$$

where N is the number of items in the clustering. The entropy is defined as

$$H(\Omega) = -\sum_k \frac{|\omega_k|}{N} log_2 \frac{|\omega_k|}{N}$$

The reference clusters consist of all relations annotated with the same relation type. The predicted clusters consist of all relations which were assigned the same relation type.

In addition to NMI, we also compute the F measure (Amigó et al., 2009). The F measure is computed as:

$$F = \sum_i \frac{|L_i|}{n} \text{max}_j\{F(L_i,C_j)\}$$

where

$$F(L_i,C_j) = \frac{2 \times Recall(L_i,C_j) \times Precision(L_i,C_j)}{Recall(L_i,C_j) + Precision(L_i,C_j)}$$

and $Precision$ is defined as:

$$Precision(C_i,L_j) = \frac{|C_i \cap L_j|}{|C_i|}$$

while $Recall$ is simply precision with the arguments swapped:

$$Recall(L,C) = Precision(C,L)$$

Table 1 shows the NMI and F measure scores for several baselines along with the RDM. Evaluation was performed on both DS1 (concept pairs having a manually annotated relation) and DS2 (all consecutive concept pairs). For DS2 we learned the models using all of the data, and evaluated on those entity pairs which had a manual relation annotated. The LDA-based model from Section 3.1 is used as one baseline. Two other baselines are K-means and Complete-Link hierarchical agglomerative clustering using TF-IDF vectors of the context and argument words (similar to Hasegawa et al. (2004)).

Method	DS1		DS2	
	NMI	F	NMI	F
Train set				
Complete-link	4.2	37.8	N/A	N/A
K-means	8.25	38.0	5.4	38.1
LDA baseline	12.8	23.0	15.6	26.2
RDM	18.2	39.1	18.1	37.4
Test set				
LDA baseline	10.0	26.1	11.5	26.3
RDM	11.8	37.7	14.0	36.4

Table 1: NMI and F measure scores for the RDM and baselines. The first two columns of numbers show the scores when evaluation is restricted to only those pairs of concepts which had a relation identified by annotators. The last two columns are the NMI and F measure scores when each method clusters all consecutive entity pairs, but is only evaluated on those with a relation identified by annotators.

Complete-link clustering did not finish on DS2 because of the large size of the data set. This highlights another advantage of the RDM. Hierarchical agglomerative clustering is quadratic in the size of the number of instances to be clustered, while the RDM's time and memory requirements both grow linearly in the number of entity pairs. The scores shown in Table 1 use the best parameterization of each model as measured by NMI. For DS1 the best LDA-based model used 15 clusters. K-means achieved the best result with 40 clusters, while the best Complete-Link clustering was obtained by using 40 clusters. The best RDM model used parameters $R = 9$ relation, $K = 15$ general word classes, and $A = 15$ argument classes. For DS2 the best number of clusters for LDA was 10, while K-means performed best with 58 clusters. The best RDM model used $R = 100$ relations, $K = 50$ general word classes, and $A = 15$ argument classes. The LDA-based approach saw an improvement when using the larger data set, however the RDM still performed the best.

To assess how well the RDM performs on unseen data we also evaluated the relations extracted by the model on the test set. Only the RDM and LDA models were evaluated as clusters produced by K-means and hierarchical clustering are valid only for the data used to generate the clusters. Generative models on the other hand can provide an estimate of the probability for each relation type on unseen text. For each model we generate 10 samples after a burn in period of 30 iterations and form clusters by assigning each pair of concepts to the relation assigned most often in the samples. The results of this evaluation are presented in Table 1. While these cluster scores are lower than those on the data used to train the models, they still show the RDM outperforming the LDA baseline model.

6 Discussion

The relation and argument clusters determined by the RDM provide a better unsupervised relation discovery method than the baselines. The RDM does this using no knowledge about syntax or semantics outside of that used to determine concepts. The analysis shows that words highly indicative of relations are detected and clustered automatically, without the need for prior annotation of relations or even the choice of a predetermined set of relation types. The discovered relations can be interpreted by a human or labeled automatically using a technique such as the one presented in Pantel and Ravichandran (2004). The fact that the discovered relations and argument classes align well with those chosen by annotators on the same data justify our assumptions about relations being present and discoverable by the way they are expressed in text. Table 1 shows that the model does not perform as well when many of the pairs of entities do not have a relation, but it still performs better than the baselines.

While the RDM relies in large part on trigger words for making clustering decisions it is also capable of including examples which do not contain any contextual words between the arguments. In addition to modeling trigger words, a joint distribution on argument semantic classes is also incorporated. This allows the model to determine a relation type even in the absence of triggers. For example, consider the entity pair "[lung cancer] [XRT]", where XRT stands for external radiation therapy. By determining the semantic classes for the arguments (lung cancer is a Problem, and XRT is a test), the set of possible relations between the arguments can be narrowed down. For instance, XRT is unlikely to be in a causal relationship with a problem, or to make

a problem worse. A further aid is the fact that the learned relationships may be specialized. For instance, there may be a learned relation type such as "Cancer treatment addresses cancer problem". In this case, seeing a type of cancer (lung cancer) and a type of cancer treatment (XRT) would be strong evidence for that type of relation, even without trigger words.

7 Conclusions

We presented a novel unsupervised approach to discovering relations in the narrative of electronic medical records. We developed a generative model which can simultaneously cluster relation trigger words as well as relation arguments. The model makes use of only the tokens found in the context of pairs of entities. Unlike many previous approaches, we assign relations to entities at the location those entities appear in text, allowing us to discover context-sensitive relations. The RDM outperforms baselines built using Latent Dirichlet Allocation and traditional clustering methods. The discovered relations can be used for a number of applications such as detecting when certain treatments were administered or determining if a necessary test has been performed. Future work will include transforming the RDM into a non-parametric model by using the Chinese Restaurant Process (CRP) (Blei et al., 2010). The CRP can be used to determine the number of relations, argument classes, and general word classes automatically.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital libraries*, pages 85–94, San Antonio, Texas, United States. ACM.

E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

David M. Blei, Thomas L. Griffiths, and Michael I. Jordan. 2010. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM*, 57(2):1–30.

T. L Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics. ACM ID: 1219008.

C. D Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to information retrieval*, volume 1. Cambridge University Press.

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Annual Meeting Association for Computational Linguistics*, volume 44, page 113.

P. Pantel and D. Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of HLT/NAACL*, volume 4, page 321–328.

Benjamin Rosenfeld and Ronen Feldman. 2007. Clustering for unsupervised relation identification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, page 411–418, New York, NY, USA. ACM. ACM ID: 1321499.

Z. Syed and E. Viegas. 2010. A hybrid approach to unsupervised relation discovery based on linguistic analysis and semantic typing. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, page 105–113.

Ozlem Uzuner, Brett South, Shuying Shen, and Scott DuVall. 2011. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *Accepted for publication.*

A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland. 2007. TextRunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, page 25–26.

Alexander Yates. 2009. Unsupervised resolution of objects and relations on the web. *Journal of Artificial Intelligence Research*, 34(1).

Random Walk Inference and Learning in A Large Scale Knowledge Base

Ni Lao
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
nlao@cs.cmu.edu

Tom Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
tom.mitchell@cs.cmu.edu

William W. Cohen
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Abstract

We consider the problem of performing learning and inference in a large scale knowledge base containing imperfect knowledge with incomplete coverage. We show that a soft inference procedure based on a combination of constrained, weighted, random walks through the knowledge base graph can be used to reliably infer new beliefs for the knowledge base. More specifically, we show that the system can learn to infer different target relations by tuning the weights associated with random walks that follow different paths through the graph, using a version of the Path Ranking Algorithm (Lao and Cohen, 2010b). We apply this approach to a knowledge base of approximately 500,000 beliefs extracted imperfectly from the web by NELL, a never-ending language learner (Carlson et al., 2010). This new system improves significantly over NELL's earlier Horn-clause learning and inference method: it obtains nearly double the precision at rank 100, and the new learning method is also applicable to many more inference tasks.

1 Introduction

Although there is a great deal of recent research on extracting knowledge from text (Agichtein and Gravano, 2000; Etzioni et al., 2005; Snow et al., 2006; Pantel and Pennacchiotti, 2006; Banko et al., 2007; Yates et al., 2007), much less progress has been made on the problem of drawing reliable inferences from this imperfectly extracted knowledge. In particular, traditional logical inference methods are too brittle to be used to make complex inferences from automatically-extracted knowledge, and probabilistic inference methods (Richardson and Domingos, 2006) suffer from scalability problems. This paper considers the problem of constructing inference methods that can scale to large knowledge bases (KB's), and that are robust to imperfect knowledge. The KB we consider is a large triple store, which can be represented as a labeled, directed graph in which each entity a is a node, each binary relation $R(a, b)$ is an edge labeled R between a and b, and unary concepts $C(a)$ are represented as an edge labeled "isa" between the node for the entity a and a node for the concept C. We present a trainable inference method that learns to infer relations by combining the results of different random walks through this graph, and show that the method achieves good scaling properties and robust inference in a KB containing over 500,000 triples extracted from the web by the NELL system (Carlson et al., 2010).

1.1 The NELL Case Study

To evaluate our approach experimentally, we study it in the context of the NELL (Never Ending Language Learning) research project, which is an effort to develop a never-ending learning system that operates 24 hours per day, for years, to continuously improve its ability to read (extract structured facts from) the web (Carlson et al., 2010). NELL began operation in January 2010. As of March 2011, NELL had built a knowledge base containing several million candidate beliefs which it had extracted from the web with varying confidence. Among these,

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 529–539,
Edinburgh, Scotland, UK, July 27–31, 2011.

NELL had fairly high confidence in approximately half a million, which we refer to as NELL's *(confident) beliefs*. NELL had lower confidence in a few million others, which we refer to as its *candidate beliefs*.

NELL is given as input an ontology that defines hundreds of categories (e.g., person, beverage, athlete, sport) and two-place typed relations among these categories (e.g., *atheletePlaysSport*(⟨athlete⟩, ⟨sport⟩)), which it must learn to extract from the web. It is also provided a set of 10 to 20 positive seed examples of each such category and relation, along with a downloaded collection of 500 million web pages from the ClueWeb2009 corpus (Callan and Hoy, 2009) as unlabeled data, and access to 100,000 queries each day to Google's search engine. Each day, NELL has two tasks: (1) to extract additional beliefs from the web to populate its growing knowledge base (KB) with instances of the categories and relations in its ontology, and (2) to learn to perform task 1 better today than it could yesterday. We can measure its learning competence by allowing it to consider the same text documents today as it did yesterday, and recording whether it extracts more beliefs, more accurately today.[1]

NELL uses a large-scale semi-supervised multi-task learning algorithm that couples the training of over 1500 different classifiers and extraction methods (see (Carlson et al., 2010)). Although many of the details of NELL's learning method are not central to this paper, two points should be noted. First, NELL is a multistrategy learning system, with components that learn from different "views" of the data (Blum and Mitchell, 1998): for instance, one view uses orthographic features of a potential entity name (like "contains capitalized words"), and another uses free-text contexts in which the noun phrase is found (e.g., "X frequently follows the bigram 'mayor of' "). Second, NELL is a bootstrapping system, which self-trains on its growing collection of confident beliefs.

1.2 Knowledge Base Inference: Horn Clauses

Although NELL has now grown a sizable knowledge base, its ability to perform inference over this knowledge base is currently very limited. At present its only inference method beyond simple inheritance involves applying first order Horn clause rules to infer new beliefs from current beliefs. For example, it may use a Horn clause such as

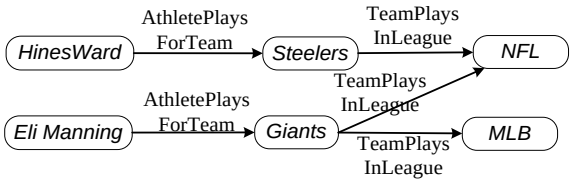


Figure 1: An example subgraph.

$$\begin{aligned} & \textit{AthletePlaysForTeam}(a, b) \\ \wedge \quad & \textit{TeamPlaysInLeague}(b, c) \\ \Rightarrow \quad & \textit{AthletePlaysInLeague(a,c)} \end{aligned} \tag{1}$$

to infer that *AthletePlaysInLeague(HinesWard,NFL)*, if it has already extracted the beliefs in the preconditions of the rule, with variables a, b and c bound to *HinesWard*, *PittsburghSteelers* and *NFL* respectively as shown in Figure 1. NELL currently has a set of approximately 600 such rules, which it has learned by data mining its knowledge base of beliefs. Each learned rule carries a conditional probability that its conclusion will hold, given that its preconditions are satisfied.

NELL learns these Horn clause rules using a variant of the FOIL algorithm (Quinlan and Cameron-Jones, 1993), henceforth N-FOIL. N-FOIL takes as input a set of positive and negative examples of a rule's consequent (e.g., $+$*AthletePlaysInLeague(HinesWard,NFL)*, $-$*AthletePlaysInLeague(HinesWard,NBA)*), and uses a "separate-and-conquer" strategy to learn a set of Horn clauses that fit the data well. Each Horn clause is learned by starting with a general rule and progressively specializing it, so that it still covers many positive examples but covers few negative examples. After a clause is learned, the examples covered by that clause are removed from the training set, and the process repeats until no positive examples remain.

Learning first-order Horn clauses is computationally expensive—not only is the search space large, but some Horn clauses can be costly to evaluate (Cohen and Page, 1995). N-FOIL uses two tricks to improve its scalability. First, it assumes that the consequent predicate is functional—e.g., that

[1] NELL's current KB is available online at `http://rtw.ml.cmu.edu`.

each *Athlete* plays in at most one *League*. This means that explicit negative examples need not be provided (Zelle et al., 1995): e.g., if *AthletePlaysInLeague(HinesWard,NFL)* is a positive example, then *AthletePlaysInLeague(HinesWard,c′)* for any other value of c' is negative. In general, this constraint guides the search algorithm toward Horn clauses that have fewer possible instantiations, and hence are less expensive to match. Second, N-FOIL uses "relational pathfinding" (Richards and Mooney, 1992) to produce general rules—i.e., the starting point for a predicate R is found by looking at positive instances $R(a, b)$ of the consequent, and finding a clause that corresponds to a bounded-length path of binary relations that link a to b. In the example above, a start clause might be the clause (1). As in FOIL, the clause is then (potentially) specialized by greedily adding additional conditions (like *ProfessionalAthlete*(a)) or by replacing variables with constants (eg, replacing c with *NFL*).

For each N-FOIL rule, an estimated conditional probability $\hat{P}(conclusion|preconditions)$ is calculated using a Dirichlet prior according to

$$\hat{P} = (N_+ + m * prior)/(N_+ + N_- + m) \quad (2)$$

where N_+ is the number of positive instances matched by this rule in the FOIL training data, N_- is the number of negative instances matched, $m = 5$ and $prior = 0.5$. As the results below show, N-FOIL generally learns a small number of high-precision inference rules. One important role of these inference rules is that they contribute to the bootstrapping procedure, as inferences made by N-FOIL increase either the number of candidate beliefs, or (if the inference is already a candidate) improve NELL's confidence in candidate beliefs.

1.3 Knowledge Base Inference: Graph Random Walks

In this paper, we consider an alternative approach, based on the Path Ranking Algorithm (PRA) of Lao and Cohen (2010b), described in detail below. PRA learns to *rank* graph nodes b relative to a query node a. PRA begins by enumerating a large set of bounded-length edge-labeled path types, similar to the initial clauses used in NELL's variant of FOIL. These path types are treated as ranking "experts", each performing a random walk through the graph, constrained to follow that sequence of edge types, and ranking nodes b by their weights in the resulting distribution. Finally, PRA combines the weights contributed by different "experts" using logistic regression to predict the probability that the relation $R(a, b)$ is satisfied.

As an example, consider a path from a to b via the sequence of edge types *isa*, isa^{-1} (the inverse of *isa*), and *AthletePlaysInLeague*, which corresponds to the Horn clause

$$\begin{aligned} & isa(a, c) \wedge isa^{-1}(c, a') \\ \wedge \quad & AthletePlaysInLeague(a', b) \\ \Rightarrow \quad & AthletePlaysInLeague(a, b) \end{aligned} \quad (3)$$

Suppose a random walk starts at a query node a (say a=*HinesWard*). If *HinesWard* is linked to the single concept node *ProfessionalAthlete* via *isa*, the walk will reach that node with probability 1 after one step. If A is the set of *ProfessionalAthlete*'s in the KB, then after two steps, the walk will have probability $1/|A|$ of being at any $a' \in A$. If L is the set of athletic leagues and $\ell \in L$, let A_ℓ be the set of athletes in league ℓ: after three steps, the walk will have probability $|A_\ell|/|A|$ of being at any point $b \in L$. In short, the ranking associated with this path gives the prior probability of a value b being an athletic league for a—which is useful as a feature in a combined ranking method, although not by itself a high-precision inference rule.

Note that the rankings produced by this "expert" will change as the knowledge base evolves—for instance, if the system learns about proportionally more soccer players than hockey players over time, then the league rankings for the path of clause (3) will change. Also, the ranking is specific to the query node a. For instance, suppose the KB contains facts which reflect the ambiguity of the team name "Giants"[2] as in Figure 1. Then the path for clause (1) above will give lower weight to $b =$ *NFL* for $a =$ *EliManning* than to $b =$ *NFL* for $a =$ *HinesWard*.

The main contribution of this paper is to introduce and evaluate PRA as an algorithm for making probabilistic inference in large KBs. Compared to Horn clause inference, the key characteristics of this new inference method are as follows:

[2]San Francisco's Major-League Baseball and New York's National Football League teams are both called the "Giants".

- The evidence in support of inferring a relation instance $R(a, b)$ is based on many existing paths between a and b in the current KB, combined using a learned logistic function.
- The confidence in an inference is sensitive to the current state of the knowledge base, and the specific entities being queried (since the paths used in the inference have these properties).
- Experimentally, the inference method yields many more moderately-confident inferences than the Horn clauses learned by N-FOIL.
- The learning and inference are more efficient than N-FOIL, in part because we can exploit efficient approximation schemes for random walks (Lao and Cohen, 2010a). The resulting inference is as fast as 10 milliseconds per query on average.

The Path Ranking Algorithm (PRA) we use is similar to that described elsewhere (Lao and Cohen, 2010b), except that to achieve efficient model learning, the paths between a and b are determined by the statistics from a population of training queries rather than enumerated completely. PRA uses random walks to generate relational features on graph data, and combine them with a logistic regression model. Compared to other relational models (e.g. FOIL, Markov Logic Networks), PRA is extremely efficient at link prediction or retrieval tasks, in which we are interested in identifying top links from a large number of candidates, instead of focusing on a particular node pair or joint inferences.

1.4 Related Work

The TextRunner system (Cafarella et al., 2006) answers list queries on a large knowledge base produced by open domain information extraction. Spreading activation is used to measure the closeness of any node to the query term nodes. This approach is similar to the random walk with restart approach which is used as a baseline in our experiment. The FactRank system (Jain and Pantel, 2010) compares different ways of constructing random walks, and combining them with extraction scores. However, the shortcoming of both approaches is that they ignore edge type information, which is important for achieving high accuracy predictions.

The HOLMES system (Schoenmackers et al., 2008) derives new assertions using a few manually written inference rules. A Markov network corresponding to the grounding of these rules to the knowledge base is constructed for each query, and then belief propagation is used for inference. In comparison, our proposed approach discovers inference rules automatically from training data.

Similarly, the Markov Logic Networks (Richardson and Domingos, 2006) are Markov networks constructed corresponding to the grounding of rules to knowledge bases. In comparison, our proposed approach is much more efficient by avoiding the harder problem of joint inferences and by leveraging efficient random walk schemes (Lao and Cohen, 2010a).

Below we describe our approach in greater detail, provide experimental evidence of its value for performing inference in NELL's knowledge base, and discuss implications of this work and directions for future research.

2 Approach

In this section, we first describe how we formulate link (relation) prediction on a knowledge base as a ranking task. Then we review the Path Ranking Algorithm (PRA) introduced by Lao and Cohen (2010b; 2010a). After that, we describe two improvements to the PRA method to make it more suitable for the task of link prediction in knowledge bases. The first improvement helps PRA deal with the large number of relations typical of large knowledge bases. The second improvement aims at improving the quality of inference by applying low variance sampling.

2.1 Learning with NELL's Knowledge Base

For each relation R in the knowledge base we train a model for the link prediction task: given a concept a, find all other concepts b which potentially have the relation $R(a, b)$. This prediction is made based on an existing knowledge base extracted imperfectly from the web. Although a model can potentially benefit from predicting multiple relations jointly, such joint inference is beyond the scope of this work.

To ensure a reasonable number of training instances, we generate labeled training example queries from 48 relations which have more than 100 instances in the knowledge base. We create two tasks for each relation—i.e., predicting b given a and predicting a given b— yielding 96 tasks in all. Each node a which has relation R in the knowledge base with any other node is treated as a training query, the actual nodes b in the knowledge base known to satisfy $R(a, b)$ are treated as labeled positive examples, and any other nodes are treated as negative examples.

2.2 Path Ranking Algorithm Review

We now review the Path Ranking Algorithm introduced by Lao and Cohen (2010b). A *relation path* P is defined as a sequence of relations $R_1 \dots R_\ell$, and in order to emphasize the types associated with each step, P can also be written as $T_0 \xrightarrow{R_1} \dots \xrightarrow{R_\ell} T_\ell$, where $T_i = range(R_i) = domain(R_{i+1})$, and we also define $domain(P) \equiv T_0$, $range(P) \equiv T_\ell$. In the experiments in this paper, there is only one type of node which we call a *concept*, which can be connected through different types of relations. In this notation, relations like "the team a certain player plays for", and "the league a certain player's team is in" can be expressed by the paths below (respectively):

$$P_1: \quad concept \xrightarrow{AtheletePlayesForTeam} concept$$

$$P_2: \quad concept \xrightarrow{AtheletePlayesForTeam} concept \xrightarrow{TeamPlaysInLeague} concept$$

For any relation path $P = R_1 \dots R_\ell$ and a seed node $s \in domain(P)$, a *path constrained random walk* defines a distribution $h_{s,P}$ recursively as follows. If P is the empty path, then define

$$h_{s,P}(e) = \begin{cases} 1, & \text{if } e = s \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

If $P = R_1 \dots R_\ell$ is nonempty, then let $P' = R_1 \dots R_{\ell-1}$, and define

$$h_{s,P}(e) = \sum_{e' \in range(P')} h_{s,P'}(e') \cdot P(e|e'; R_\ell), \quad (5)$$

where $P(e|e'; R_\ell) = \frac{R_\ell(e',e)}{|R_\ell(e',\cdot)|}$ is the probability of reaching node e from node e' with a one step random walk with edge type R_ℓ. $R(e', e)$ indicates whether there exists an edge with type R that connect e' to e.

More generally, given a set of paths $P_1, \dots, P_n$, one could treat each $h_{s,P_i}(e)$ as a *path feature* for the node e, and rank nodes by a linear model

$$\theta_1 h_{s,P_1}(e) + \theta_2 h_{s,P_2}(e) + \dots \theta_n h_{s,P_n}(e)$$

where θ_i are appropriate weights for the paths. This gives a ranking of nodes e related to the query node s by the following scoring function

$$score(e; s) = \sum_{P \in \mathbf{P}_\ell} h_{s,P}(e)\theta_P, \quad (6)$$

where $\mathbf{P}_\ell$ is the set of relation paths with length $\leq \ell$.

Given a relation R and a set of node pairs $\{(s_i, t_i)\}$ for which we know whether $R(s_i, t_i)$ is true or not, we can construct a training dataset $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}_i$ is a vector of all the path features for the pair (s_i, t_i)—i.e., the j-th component of $\mathbf{x}_i$ is $h_{s_i,P_j}(t_i)$, and where y_i is a boolean variable indicating whether $R(s_i, t_i)$ is true. We then train a logistic function to predict the conditional probability $P(y|\mathbf{x}; \theta)$. The parameter vector θ is estimated by maximizing a regularized form of the conditional likelihood of y given $\mathbf{x}$. In particular, we maximize the objective function

$$O(\theta) = \sum_i o_i(\theta) - \lambda_1|\theta|_1 - \lambda_2|\theta|_2, \quad (7)$$

where λ_1 controls L_1-regularization to help structure selection, and λ_2 controls L_2-regularization to prevent overfitting. $o_i(\theta)$ is the per-instance weighted log conditional likelihood given by

$$o_i(\theta) = w_i[y_i \ln p_i + (1 - y_i) \ln(1 - p_i)], \quad (8)$$

where p_i is the predicted probability $p(y_i = 1|\mathbf{x}_i; \theta) = \frac{\exp(\theta^T \mathbf{x}_i)}{1+\exp(\theta^T \mathbf{x}_i)}$, and w_i is an importance weight to each example. A biased sampling procedure selects only a small subset of negative samples to be included in the objective (see (Lao and Cohen, 2010b) for detail).

2.3 Data-Driven Path Finding

In prior work with PRA, $\mathbf{P}_\ell$ was defined as all relation paths of length at most ℓ. When the number of edge types is small, one can generate $\mathbf{P}_\ell$ by

Table 1: Number of paths in PRA models of maximum path length 3 and 4. Averaged over 96 tasks.

	ℓ=3	ℓ=4
all paths up to length L	$15,376$	$1,906,624$
+query support$\geq \alpha = 0.01$	522	5016
+ever reach a target entity	136	792
+L_1 regularization	63	271

Table 2: Comparing PRA with RWR models. MRRs and training times are averaged over 96 tasks.

	ℓ=2		ℓ=3	
	MRR	**Time**	**MRR**	**Time**
RWR(no train)	0.271		0.456	
RWR	0.280	3.7s	0.471	9.2s
PRA	0.307	5.7s	0.516	15.4s

enumeration; however, for domains with a large number of edge types (e.g., a knowledge base), it is impractical to enumerate all possible relation paths even for small ℓ. For instance, if the number of edge types related to each node type is 100, even the number of length three paths types easily reaches millions. For other domains like parsed natural language sentences, useful relation paths can be as long as ten relations (Minkov and Cohen, 2008). In this case, even with smaller number of possible edge types, the total number of relation paths is still too large for systematic enumeration.

In order to apply PRA to these domains, we modify the path generation procedure in PRA to produce only relation paths which are potentially useful for the task. Define a query s to be *supporting* a path P if $h_{s,P}(e) \neq 0$ for any entity e. We require that any path node created during path finding needs to be supported by at least a fraction α of the training queries s_i, as well as being of length no more than ℓ (In the experiments, we set $\alpha = 0.01$) We also require that in order for a relation path to be included in the PRA model, it must retrieve at least one target entity t_i in the training set. As we can see from Table 1, together these two constraints dramatically reduce the number of relation paths that need to be considered, relative to systematically enumerating all possible relation paths. L1 regularization reduces the size of the model even more.

The idea of finding paths that connects nodes in a graph is not new. It has been embodied previously in first-order learning systems (Richards and Mooney, 1992) as well as N-FOIL, and relational database searching systems (Bhalotia et al., 2002). These approaches consider a single query during path finding. In comparison, the data-driven path finding method we described here uses statistics from a population of queries, and therefore can potentially determine the importance of a path more reliably.

2.4 Low-Variance Sampling

Lao and Cohen (2010a) previously showed that sampling techniques like finger printing and particle filtering can significantly speedup random walk without sacrificing retrieval quality. However, the sampling procedures can induce a loss of diversity in the particle population. For example, consider a node in the graph with just two out links with equal weights, and suppose we are required to generate two walkers starting from this node. A disappointing result is that with 50 percent chance both walkers will follow the same branch, and leave the other branch with no probability mass.

To overcome this problem, we apply a technique called Low-Variance Sampling (LVS) (Thrun et al., 2005), which is commonly used in robotics to improve the quality of sampling. Instead of generating independent samples from a distribution, LVS uses a single random number to generate all samples, which are evenly distributed across the whole distribution. Note that given a distribution $P(x)$, any number r in $[0, 1]$ points to exactly one x value, namely $x = \arg\min_j \sum_{m=1..j} P(m) \leq r$. Suppose we want to generate M samples from $P(x)$. LVS first generates a random number r in the interval $[0, M^{-1}]$. Then LVS repeatedly adds the fixed amount M^{-1} to r and chooses x values corresponding to the resulting numbers.

3 Results

This section reports empirical results of applying random walk inference to NELL's knowledge base after the 165th iteration of its learning process. We first investigate PRA's behavior by cross validation on the training queries. Then we compare PRA and N-FOIL's ability to reliably infer new beliefs, by leveraging the Amazon Mechanical Turk service.

3.1 Cross Validation on the Training Queries

Random Walk with Restart (RWR) (also called personalized PageRank (Haveliwala, 2002)) is a general-purpose graph proximity measure which has been shown to be fairly successful for many types of tasks. We compare PRA to two versions of RWR on the 96 tasks of link prediction with NELL's knowledge base. The two baseline methods are an untrained RWR model and a trained RWR model as described by Lao and Cohen (2010b). (In brief, in the trained RWR model, the walker will probabilistically prefer to follow edges associated with different labels, where the weight for each edge label is chosen to minimize a loss function, such as Equation 7. In the untrained model, edge weights are uniform.) We explored a range of values for the regularization parameters L_1 and L_2 using cross validation on the training data, and we fix both L_1 and L_2 parameters to 0.001 for all tasks. The maximum path length is fixed to 3.[3]

Table 2 compares the three methods using 5-fold cross validation and the Mean Reciprocal Rank (MRR)[4] measure, which is defined as the inverse rank of the highest ranked relevant result in a set of results. If the the first returned result is relevant, then MRR is 1.0, otherwise, it is smaller than 1.0. Supervised training can significantly improve retrieval quality (p-value=9×10^{-8} comparing untrained and trained RWR), and leveraging path information can produce further improvement (p-value=4×10^{-4} comparing trained RWR with PRA). The average training time for a predicate is only a few seconds.

We also investigate the effect of low-variance sampling on the quality of prediction. Figure 2 compares independent and low variance sampling when applied to finger printing and particle filtering (Lao and Cohen, 2010a). The horizontal axis corresponds to the speedup of random walk compared with exact inference, and the vertical axis measures the quality of prediction by MRR with three fold cross validation on the training query set. Low-variance

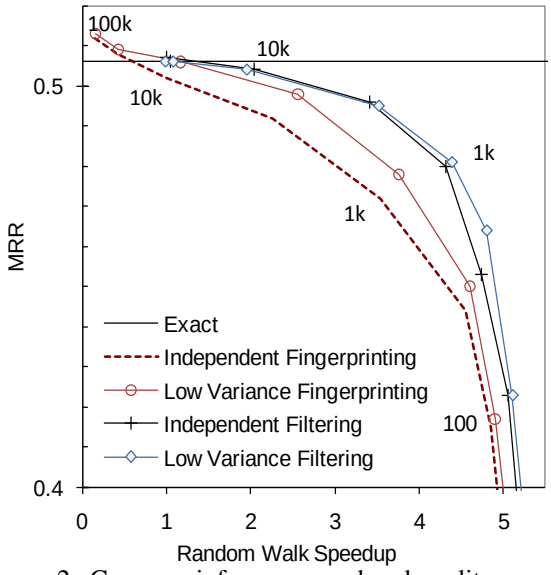


Figure 2: Compare inference speed and quality over 96 tasks. The speedup is relative to exact inference, which is on average 23ms per query.

sampling can improve prediction for both finger printing and particle filtering. The numbers on the curves indicate the number of particles (or walkers). When using a large number of particles, the particle filtering methods converge to the exact inference. Interestingly, when using a large number of walkers, the finger printing methods produce even better prediction quality than exact inference. Lao and Cohen noticed a similar improvement on retrieval tasks, and conjectured that it is because the sampling inference imposes a regularization penalty on longer relation paths (2010a).

3.2 Evaluation by Mechanical Turk

The cross-validation result above assumes that the knowledge base is complete and correct, which we know to be untrue. To accurately compare PRA and N-FOIL's ability to reliably infer new beliefs from an imperfect knowledge base, we use human assessments obtained from Amazon Mechanical Turk. To limit labeling costs, and since our goal is to improve the performance of NELL, we do not include RWR-based approaches in this comparison. Among all the 24 functional predicates, N-FOIL discovers confident rules for 8 of them (it produces no result for the other 16 predicates). Therefore, we compare the quality of PRA to N-FOIL on these 8 predicates only. Among all the 72 non-functional predicates—which

[3] Results with maximum length 4 are not reported here. Generally models with length 4 paths produce slightly better results, but are 4-5 times slower to train

[4] For a set of queries Q, $MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{rank of the first correct answer for } q}$

Table 3: The top two weighted PRA paths for tasks on which N-FOIL discovers confident rules. c stands for *concept*.

ID	PRA Path (Comment)
	athletePlaysForTeam
1	c $\xrightarrow{\text{athletePlaysInLeague}}$ c $\xrightarrow{\text{leaguePlayers}}$ c $\xrightarrow{\text{athletePlaysForTeam}}$ c (teams with many players in the athlete's league)
2	c $\xrightarrow{\text{athletePlaysInLeague}}$ c $\xrightarrow{\text{leagueTeams}}$ c $\xrightarrow{\text{teamAgainstTeam}}$ c (teams that play against many teams in the athlete's league)
	athletePlaysInLeague
3	c $\xrightarrow{\text{athletePlaysSport}}$ c $\xrightarrow{\text{players}}$ c $\xrightarrow{\text{athletePlaysInLeague}}$ c (the league that players of a certain sport belong to)
4	c $\xrightarrow{\text{isa}}$ c $\xrightarrow{\text{isa}^{-1}}$ c $\xrightarrow{\text{athletePlaysInLeague}}$ c (popular leagues with many players)
	athletePlaysSport
5	c $\xrightarrow{\text{isa}}$ c $\xrightarrow{\text{isa}^{-1}}$ c $\xrightarrow{\text{athletePlaysSport}}$ c (popular sports of all the athletes)
6	c $\xrightarrow{\text{athletePlaysInLeague}}$ c $\xrightarrow{\text{superpartOfOrganization}}$ c $\xrightarrow{\text{teamPlaysSport}}$ c (popular sports of a certain league)
	stadiumLocatedInCity
7	c $\xrightarrow{\text{stadiumHomeTeam}}$ c $\xrightarrow{\text{teamHomeStadium}}$ c $\xrightarrow{\text{stadiumLocatedInCity}}$ c (city of the stadium with the same team)
8	c $\xrightarrow{\text{latitudeLongitude}}$ c $\xrightarrow{\text{latitudeLongitudeOf}}$ c $\xrightarrow{\text{stadiumLocatedInCity}}$ c (city of the stadium with the same location)
	teamHomeStadium
9	c $\xrightarrow{\text{teamPlaysInCity}}$ c $\xrightarrow{\text{cityStadiums}}$ c (stadiums located in the same city with the query team)
10	c $\xrightarrow{\text{teamMember}}$ c $\xrightarrow{\text{athletePlaysForTeam}}$ c $\xrightarrow{\text{teamHomeStadium}}$ c (home stadium of teams which share players with the query)
	teamPlaysInCity
11	c $\xrightarrow{\text{teamHomeStadium}}$ c $\xrightarrow{\text{stadiumLocatedInCity}}$ c (city of the team's home stadium)
12	c $\xrightarrow{\text{teamHomeStadium}}$ c $\xrightarrow{\text{stadiumHomeTeam}}$ c $\xrightarrow{\text{teamPlaysInCity}}$ c (city of teams with the same home stadium as the query)
	teamPlaysInLeague
13	c $\xrightarrow{\text{teamPlaysSport}}$ c $\xrightarrow{\text{players}}$ c $\xrightarrow{\text{athletePlaysInLeague}}$ c (the league that the query team's members belong to)
14	c $\xrightarrow{\text{teamPlaysAgainstTeam}}$ c $\xrightarrow{\text{teamPlaysInLeague}}$ c (the league that the query team's competing team belongs to)
	teamPlaysSport
15	c $\xrightarrow{\text{isa}}$ c $\xrightarrow{\text{isa}^{-1}}$ c $\xrightarrow{\text{teamPlaysSport}}$ c (sports played by many teams)
16	c $\xrightarrow{\text{teamPlaysInLeague}}$ c $\xrightarrow{\text{leagueTeams}}$ c $\xrightarrow{\text{teamPlaysSport}}$ c (the sport played by other teams in the league)

Table 4: Amazon Mechanical Turk evaluation for the promoted knowledge. Using paired t-test at task level, PRA is not statistically different from N-FOIL for p@10 (p-value=0.3), but is significantly better for p@100 (p-value=0.003)

		PRA				**N-FOIL**				
Task	$P_{majority}$	**#Paths**	**p@10**	**p@100**	**p@1000**	**#Rules**	**#Query**	**p@10**	**p@100**	**p@1000**
athletePlaysForTeam	0.07	125	0.4	0.46	0.66	1(+1)	7	0.6	0.08	0.01
athletePlaysInLeague	0.60	15	1.0	0.84	0.80	3(+30)	332	0.9	0.80	0.24
athletePlaysSport	0.73	34	1.0	0.78	0.70	2(+30)	224	1.0	0.82	0.18
stadiumLocatedInCity	0.05	18	0.9	0.62	0.54	1(+0)	25	0.7	0.16	0.00
teamHomeStadium	0.02	66	0.3	0.48	0.34	1(+0)	2	0.2	0.02	0.00
teamPlaysInCity	0.10	29	1.0	0.86	0.62	1(+0)	60	0.9	0.56	0.06
teamPlaysInLeague	0.26	36	1.0	0.70	0.64	4(+151)	30	0.9	0.18	0.02
teamPlaysSport	0.42	21	0.7	0.60	0.62	4(+86)	48	0.9	0.42	0.02
average	0.28	43	0.79	0.668	0.615		91	0.76	0.38	0.07
teamMember	0.01	203	0.8	0.64	0.48					
companiesHeadquarteredIn	0.05	42	0.6	0.54	0.60					
publicationJournalist	0.02	25	0.7	0.70	0.64					
producedBy	0.19	13	0.5	0.58	0.68	N-FOIL does not produce results				
competesWith	0.19	74	0.6	0.56	0.72	for non-functional predicates				
hasOfficeInCity	0.03	262	0.9	0.84	0.60					
teamWonTrophy	0.24	56	0.5	0.50	0.46					
worksFor	0.13	62	0.6	0.60	0.74					
average	0.11	92	0.650	0.620	0.615					

N-FOIL cannot be applied to—PRA exhibits a wide range of performance in cross-validation. The are 43 tasks for which PRA obtains MRR higher than 0.4 and builds a model with more than 10 path features. We randomly sampled 8 of these predicates to be evaluated by Amazon Mechanical Turk.

Table 3 shows the top two weighted PRA features for each task on which N-FOIL can successfully learn rules. These PRA rules can be categorized into broad coverage rules which behave like priors over correct answers (e.g. 1-2, 4-6, 15), accurate rules which leverage specific relation sequences (e.g. 9, 11, 14), rules which leverage information about the synonyms of the query node (e.g. 7-8, 10, 12), and rules which leverage information from a local neighborhood of the query node (e.g. 3, 12-13, 16). The synonym paths are useful, because an entity may have multiple names on the web. We find that all 17 general rules (no specialization) learned by N-FOIL can be expressed as length two relation paths such as path 11. In comparison, PRA explores a feature space with many length three paths.

For each relation R to be evaluated, we generate test queries s which belong to $domain(R)$. Queries which appear in the training set are excluded. For each query node s, we applied a trained model (either PRA or N-FOIL) to generate a ranked list of candidate t nodes. For PRA, the candidates are sorted by their scores as in Eq. (6). For N-FOIL, the candidates are sorted by the estimated accuracies of the rules as in Eq. (2) (which generate the candidates). Since there are about 7 thousand (and 13 thousand) test queries s for each functional (and non-functional) predicate R, and there are (potentially) thousands of candidates t returned for each query s, we cannot evaluate all candidates of all queries. Therefore, we first sort the queries s for each predicate R by the scores of their top ranked candidate t in descending order, and then calculate precisions at top 10, 100 and 1000 positions for the list of result $R(s^{R,1}, t_1^{R,1}), R(s^{R,2}, t_1^{R,2}), ...$, where $s^{R,1}$ is the first query for predicate R, $t_1^{R,1}$ is its first candidate, $s^{R,2}$ is the second query for predicate R, $t_1^{R,2}$ is its first candidate, so on and so forth. To reduce the labeling load, we judge all top 10 queries for each predicate, but randomly sample 50 out of the top 100, and randomly sample 50 out of the top 1000. Each belief is evaluated by 5 workers at Mechanical Turk, who are given assertions like "*Hines Ward* plays for the team *Steelers*", as well as Google search links for each entity, and the combination of both entities. Statistics shows that the workers spend on average 25 seconds to judge each belief. We also remove some workers' judgments which are obviously incorrect[5]. We sampled 100 beliefs, and compared their voted result to gold-standard labels produced by one author of this paper. Table 5 shows that 74% of the time the workers' voted result agrees with our judgement.

Table 5: Comparing Mechanical Turk workers' voted assessments with our gold standard labels based on 100 samples.

	AMT=F	AMT=T
Gold=F	25%	15%
Gold=T	11%	49%

Table 4 shows the evaluation result. The $P_{majority}$ column shows for each predicate the accuracy achieved by the majority prediction: given a query $R(a, ?)$, predict the b that most often satisfies R over all possible a in the knowledge base. Thus, the higher $P_{majority}$ is, the simpler the task. Predicting the functional predicates is generally easier predicting the non-functional predicates. The #Query column shows the number of queries on which N-FOIL is able to match any of its rules, and hence produce a candidate belief. For most predicates, N-FOIL is only able to produce results for at most a few hundred queries. In comparison, PRA is able to produce results for 6,599 queries on average for each functional predicate, and 12,519 queries on average for each non-functional predicate. Although the precision at 10 (p@10) of N-FOIL is comparable to that of PRA, precision at 10 and at 1000 (p@100 and p@1000) are much lower[6].

The #Path column shows the number of paths learned by PRA, and the #Rule column shows the number of rules learned by N-FOIL, with the numbers before brackets correspond to unspecialized rules, and the numbers in brackets correspond to

[5]Certain workers label all the questions with the same answer

[6]If a method makes k predictions, and $k < n$, then p@n is the number correct out of the k predictions, divided by n

specialized rules. Generally, specialized rules have much smaller recall than unspecialized rules. Therefore, the PRA approach achieves high recall partially by combining a large number of unspecialized paths, which correspond to unspecialized rules. However, learning more accurate specialized paths is part of our future work.

A significant advantage of PRA over N-FOIL is that it can be applied to non-functional predicates. The last eight rows of Table 4 show PRA's performance on eight of these predicates. Compared to the result on functional predicates, precisions at 10 and at 100 of non-functional predicates are slightly lower, but precisions at 1000 are comparable. We note that for some predicates precision at 1000 is better than at 100. After some investigation we found that for many relations, the top portion of the result list is more diverse: i.e. showing products produced by different companies, journalist working at different publications. While the lower half of the result list is more homogeneous: i.e. showing relations concentrated on one or two companies/publications. On the other hand, through the process of labeling the Mechanical Turk workers seem to build up a prior about which company/publication is likely to have correct beliefs, and their judgments are positively biased towards these companies/publications. These two factors combined together result in positive bias towards the lower portion of the result list. In future work we hope to design a labeling strategy which avoids this bias.

4 Conclusions and Future Work

We have shown that a soft inference procedure based on a combination of constrained, weighted, random walks through the knowledge base graph can be used to reliably infer new beliefs for the knowledge base. We applied this approach to a knowledge base of approximately 500,000 beliefs extracted imperfectly from the web by NELL. This new system improves significantly over NELL's earlier Horn-clause learning and inference method: it obtains nearly double the precision at rank 100. The inference and learning are both very efficient—our experiment shows that the inference time is as fast as 10 milliseconds per query on average, and the training for a predicate takes only a few seconds.

There are several prominent directions for future work. First, inference starting from both the query nodes and target nodes (Richards and Mooney, 1992) can be much more efficient in discovering long paths than just inference from the query nodes. Second, inference starting from the target nodes of training queries is a potential way to discover specialized paths (with grounded nodes). Third, generalizing inference paths to inference trees or graphs can produce more expressive random walk inference models. Overall, we believe that random walk is a promising way to scale up relational learning to domains with very large data sets.

Acknowledgments

This work was supported by NIH under grant R01GM081293, by NSF under grant IIS0811562, by DARPA under awards FA8750-08-1-0009 and AF8750-09-C-0179, and by a gift from Google. We thank Geoffrey J. Gordon for the suggestion of applying low variance sampling to random walk inference. We also thank Bryan Kisiel for help with the NELL system.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries - DL '00*, pages 85–94, New York, New York, USA. ACM Press.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *IJCAI*, pages 2670–2676.

Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, and S. Sudarshan. 2002. Keyword searching and browsing in databases using banks. *ICDE*, pages 431–440.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98*, pages 92–100, New York, New York, USA. ACM Press.

MJ Cafarella, M Banko, and O Etzioni. 2006. Relational Web Search. In *WWW*.

Jamie Callan and Mark Hoy. 2009. Clueweb09 data set. *http://boston.lti.cs.cmu.edu/Data/clueweb09/*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M.

Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *AAAI*.

William W. Cohen and David Page. 1995. Polynomial learnability and inductive logic programming: Methods and results. *New Generation Comput.*, 13(3&4):369–409.

Oren Etzioni, Michael Cafarella, Doug Downey, Anamaria Popescu, Tal Shaked, Stephen Soderl, Daniel S. Weld, and Er Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165:91–134.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *WWW*, pages 517–526.

Alpa Jain and Patrick Pantel. 2010. Factrank: Random walks on a web of facts. In *COLING*, pages 501–509.

Ni Lao and William W. Cohen. 2010a. Fast query execution for retrieval models based on path-constrained random walks. *KDD*.

Ni Lao and William W. Cohen. 2010b. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*.

Einat Minkov and William W. Cohen. 2008. Learning graph walk based similarity measures for parsed text. *EMNLP*, pages 907–916.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *ACL*.

J. Ross Quinlan and R. Mike Cameron-Jones. 1993. FOIL: A Midterm Report. In *ECML*, pages 3–20.

Bradley L. Richards and Raymond J. Mooney. 1992. Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 50–55, San Jose, CA, July.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*.

Stefan Schoenmackers, Oren Etzioni, and Daniel S. Weld. 2008. Scaling Textual Inference to the Web. In *EMNLP*, pages 79–88.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic Taxonomy Induction from Heterogenous Evidence. In *ACL*.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. 2005. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael J. Cafarella, Oren Etzioni, and Stephen Soderland. 2007. TextRunner: Open Information Extraction on the Web. In *HLT-NAACL (Demonstrations)*, pages 25–26.

John M. Zelle, Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1995. Inducing logic programs without explicit negative examples. In *Proceedings of the Fifth International Workshop on Inductive Logic Programming (ILP-95)*, pages 403–416, Leuven, Belgium.

Exploring Supervised LDA Models for Assigning Attributes to Adjective-Noun Phrases

Matthias Hartung and **Anette Frank**
Computational Linguistics Department
Heidelberg University
{hartung,frank}@cl.uni-heidelberg.de

Abstract

This paper introduces an attribute selection task as a way to characterize the inherent meaning of property-denoting adjectives in adjective-noun phrases, such as e.g. *hot* in *hot summer* denoting the attribute TEMPERATURE, rather than TASTE. We formulate this task in a vector space model that represents adjectives and nouns as vectors in a semantic space defined over possible attributes. The vectors incorporate latent semantic information obtained from two variants of LDA topic models. Our LDA models outperform previous approaches on a small set of 10 attributes with considerable gains on sparse representations, which highlights the strong smoothing power of LDA models. For the first time, we extend the attribute selection task to a new data set with more than 200 classes. We observe that large-scale attribute selection is a hard problem, but a subset of attributes performs robustly on the large scale as well. Again, the LDA models outperform the VSM baseline.

1 Introduction

Corpus-based statistical modeling of semantics is gaining increased attention in computational linguistics. This field of research includes distributional vector space models (VSMs), i.e., models that represent the semantics of words or phrases as vectors over high-dimensional cooccurrence data (Turney and Pantel, 2010; Baroni and Lenci, 2010, i.a.), as well as latent variable models (LVMs) which aggregate distributional observations in 'hidden', or latent variables, thereby reducing the dimensionality of the data. An example of the latter are topic models (Blei et al., 2003), which have recently been applied to modeling selectional preferences of verbs (Ritter et al., 2010; Ó Séaghdha, 2010), or word sense disambiguation (Li et al., 2010).

A topic that is increasingly studied in distributional semantics is the semantics of adjectives, both in isolation (Almuhareb, 2006) and in compositional adjective-noun phrases (Hartung and Frank, 2010; Guevara, 2010; Baroni and Zamparelli, 2010).

In this paper, we propose a new approach to a problem we denote as *attribute selection*: The task is to predict the hidden attribute meaning expressed by a property-denoting adjective in composition with a noun. The adjective *hot*, e.g., may denote attributes such as TEMPERATURE, TASTE or EMOTIONALITY. These adjective meanings can be combined with nouns such as *tea, soup* or *debate*, which can be characterized in terms of attributes as well. The goal of the task is to determine the hidden attribute meaning predicated over the noun in a given adjective-noun phrase, as illustrated in (1).

(1) a. a $\text{hot}_{\text{value}}$ $\text{summer}_{\text{concept}}$
b. TEMPERATURE(*summer*) = *hot*

It is by way of the composition of adjective and noun that specific attributes are selected from the adjective's space of possible attribute meanings, and typically lead to a disambiguation of the adjective and possibly the noun. Hartung and Frank (2010) were the first to model this insight in a VSM by representing the meaning of adjectives and nouns in semantic vectors defined over attributes. The meaning of adjective-noun phrases is computed by means of

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 540–551,
Edinburgh, Scotland, UK, July 27–31, 2011.

	COLOR	DIRECTION	DURATION	SHAPE	SIZE	SMELL	SPEED	TASTE	TEMPERATURE	WEIGHT
$\vec{e}$	1	1	0	1	45	0	4	0	0	21
$\vec{b}$	14	38	2	20	26	0	45	0	0	20
$\vec{e} \times \vec{b}$	14	38	0	20	**1170**	0	180	0	0	420
$\vec{e} + \vec{b}$	15	39	2	21	**71**	0	49	0	0	41

Figure 1: Vectors for *enormous* ($\vec{e}$) and *ball* ($\vec{b}$)

vector composition, such that the 'hidden' attribute meaning of the phrase can be 'selected' as a prominent component from the composed vector. This is illustrated in Fig. 1 for the adjective *enormous* ($\vec{e}$) in combination with the noun *ball* ($\vec{b}$), with alternative composition operations: vector multiplication ($\times$) and addition ($+$).[1] Both yield SIZE as the most prominent component in the composed vector.

In the present paper we offer a new approach to this formalization of the compositional meaning of adjectives and nouns that owes to both distributional VSMs and LVMs. Through this combination, we attempt to improve on earlier work in Almuhareb (2006) and Hartung and Frank (2010), which are both embedded in a purely distributional setting.

Specifically, we use Latent Dirichlet Allocation (LDA; Blei et al. (2003)) to train an attribute model that captures semantic information encoded in adjectives and nouns independently of one another. Following Hartung and Frank (2010), this model is embedded into a VSM that employs vector composition to combine the meaning of adjectives and nouns. We present two variants of LDA that differ in the way attributes are associated with the induced LDA topics: Controled LDA (C-LDA) and Labeled LDA (L-LDA; Ramage et al. (2009)). Both will be presented in detail in Section 3.

Our aims in this paper are two-fold: (i) We investigate LDA as a modeling framework in the attribute selection task, as its use of topics as latent variables may alleviate inherent sparsity problems faced by prior work using pattern-based (Almuhareb, 2006) or vector space models (Hartung and Frank, 2010). (ii) While these prior approaches were restricted to a confined set of 10 attributes, we will we apply our models on a much larger space of attributes, to probe their capacity on a more realistic data set.

The remainder of this paper is divided as follows. Section 2 reviews related work on distributional models of adjective semantics, and introduces the two frameworks in which we ground our approach: LVMs and VSMs. In Section 3 we introduce two LDA models for attribute selection: C-LDA and L-LDA. Section 4 describes the settings for two experiments: In the first experiment, we perform attribute selection confined to a space of 10 attributes to compare against prior work. In the second setting we perform attribute selection on a large scale, using 206 attributes. Section 5 presents and discusses the results. Section 6 concludes.

2 Related Work

Distributional models of adjective semantics. Almuhareb (2006) aims at capturing the relationship between adjectives and attributes based on lexico-syntactic patterns, such as *the ATTR of the * is ADJ*. Apart from inherent sparsity issues, his approach does not account for the compositional nature of the problem, as the contextual information contributed by a noun is neglected: For instance, his model is unable to predict that *hot* is unlikely to denote TASTE in the context of *summer*, other than in *hot meal*.

Compositionality of adjective-noun phrases and how it can be adequately modeled in VSMs is the main concern in Baroni and Zamparelli (2010) and Guevara (2010), who are in search of the best composition operator for combining adjective with noun meanings. While these works adhere to a purely latent representation of meaning, Hartung and Frank (2010) include attributes as symbolic 'hidden' meanings of adjectives, nouns and adjective-noun phrases in a distributional VSM.

Finally, a large body of work dealing with compositionality in distributional frameworks is not confined to the special case of adjective-noun composition (Mitchell and Lapata (2008), Rudolph and Giesbrecht (2010), i.a.). All these approaches regard composition as a process combining vectors (or matrices, resp.) to yield a new, contextualized vector representation within the same semantic space.

Latent Dirichlet Allocation, aka. Topic Models (TMs). LDA is a generative probabilistic model

[1] The figure is adopted from the distributional setting of Hartung and Frank (2010), with component values defined by pattern frequency counts for the chosen attribute nouns.

for document collections. Each document is represented as a mixture over latent *topics*, where each topic is a probability distribution over words (Blei et al., 2003). These topics can be used as dense features for, e.g., document clustering. Depending on the number of topics, which has to be pre-specified, the dimensionality of the document representation can be considerably reduced in comparison to simple bag-of-words models. The remainder of this paper will assume some familiarity with LDA and the LDA terminology as introduced in Blei et al. (2003).

Recent work investigates ways of accommodating supervision with LDA, e.g. supervised topic models (Blei and McAuliffe, 2007), Labeled LDA (L-LDA) (Ramage et al., 2009) or DiscLDA (Lacoste-Julien et al., 2008). We will discuss L-LDA in Section 3.

Distributional VSMs and TMs. The idea to integrate topic models and VSMs goes back to Mitchell and Lapata (2009) who build a distributional model with dimensions set to topics over bag-of-words features. In their setting, LDA merely serves the purpose of dimensionality reduction, whereas our particular motivation is to use topics as probabilistic indicators for the prediction of attributes as semantic target categories in adjective-noun composition. Mitchell and Lapata (2010) compare VSMs defined over bags of context words vs. latent topics in a similarity judgement task. Their results indicate that a multiplicative setting works best for vector composition in word-based models, while vector addition is better suited for topic vectors.

3 Topic Models for Attribute Selection

3.1 Using LDA for modeling lexical semantics

Recently, LDA has been used for problems in lexical semantics, where the primary goal is not document modeling but the induction of semantic knowledge from high-dimensional co-occurrence data. Ritter et al. (2010) and Ó Séaghdha (2010) model selectional restrictions of verbs by inducing topic distributions that characterize 'mixtures of topics' observed in verb argument positions. As a basis for LDA modeling, they collect *pseudo-documents*, i.e. bags of words that co-occur in syntactic argument positions.

We apply a similar idea to the attribute selection problem: we collect pseudo-documents that characterize attributes by adjectives and nouns that co-occur with the attribute nouns in local contextual relations. The topic distributions obtained from fitting an LDA model to the collection of these pseudo-documents can then be injected into semantic vector representations for adjectives and nouns.

In its original statement, LDA is a fully unsupervised process (apart from the desired number of topics which has to be specified in advance) that estimates topic distributions over documents θ_d and topic-word distributions ϕ_t with topics represented as latent variables. Estimating these parameters on a document collection yields *topic proportions* $P(t|d)$ and topic distributions $P(w|t)$ that can be used to compute a smooth distribution $P(w|d)$ as in (2), where t denotes a latent topic, w a word and d a document in the corpus.

$$P(w|d) = \sum_t P(w|t)P(t|d) \qquad (2)$$

Being designed for exploratory rather than discriminative analysis, LDA does not intend conditioning of words or topics on external categories. That is, the resulting topics cannot be related to previously defined target categories. For attribute selection, the LDA-inferred topics need to be linked to semantic attributes. Therefore, we apply two extensions of standard LDA that are capable of taking supervised category information into account, either implicitly or directly, by including an additional observable variable into the generative process.

In general, LVMs can be expected to overcome sparsity issues that are frequently encountered in distributional models. This positive smoothing effect is achieved by marginalization over the latent variables (cf. Prescher et al. (2000)). For instance, it is unlikely to observe a dependency path linking the adjective *mature* to the attribute MATURITY. Such a relation is more likely for *young*, for example. If *young* co-occurs with *mature* in a different pseudo-document (AGE might be a candidate), this results in a situation where (i) *young* and *mature* share one or more latent topics and (ii) the topic proportions for the attributes MATURITY and AGE will become similar to the extent of common words in their pseudo-documents. Consequently, the final attribute model is expected to assign a (small) positive probability to the relation between *mature* and MATURITY without observing it in the training data.

3.2 Controled LDA

The generative story behind C-LDA is equivalent to standard LDA. However, the collection of pseudo-documents used as input to C-LDA is structured in a controled way such that each document conveys semantic information that specifically characterizes the individual categories of interest (attributes, in our case). In line with the distributional hypothesis (Harris, 1968), we consider the pseudo-documents constructed in this way as distributional fingerprints of the meaning of the corresponding attribute.

The contents of the pseudo-documents are selected along syntactic dependency paths linking each attribute noun to meaningful context words (adjectives and nouns).[2] A corpus consisting of the two sentences in (3), e.g., yields a pseudo-document for the attribute noun SPEED containing *car* and *fast*.

(3) What is the speed of this car? The machine runs at a very fast speed.

Though we are ultimately interested in triples of attributes, adjectives and nouns that define the compositional semantics of adjective-noun phrases (cf. (1)), C-LDA is only exposed to binary tuples between attributes and adjectives or nouns, respectively. This is in line with Hartung and Frank (2010), who obtained substantial performance improvements by splitting the ternary relation into two binary relations.

Presenting LDA with pseudo-documents that characterize individual target attributes imports supervision into the LDA process in two respects: the estimated topic proportions $P(t|d)$ will be highly attribute-specific, and similarly so for the topic distributions $P(w|t)$. This makes the model more expressive for the ultimate labeling task. Moreover, since C-LDA collects pseudo-documents focused on individual target attributes, we are able to link external categories to the generative process by heuristically labeling pseudo-documents with their respective attribute as target category. Thus, we approximate $P(w|a)$, the probability of a word given an attribute, by $P(w|d)$ as obtained from LDA:

[2]The dependency paths, together with the set of attribute nouns of interest, have to be manually specified. See the supplementary material for the full list of dependency paths used.

1 For each topic $k \in \{1, \ldots, K\}$:
2 Generate $\beta_k = (\beta_{k,1}, \ldots, \beta_{k,V})^T \sim Dir(\cdot \mid \eta)$
3 For each document d:
4 For each topic $k \in \{1, \ldots, K\}$
5 Generate $\Lambda_k^{(d)} \in \{0, 1\} \sim Bernoulli(\cdot \mid \Phi_k)$
6 Generate $\alpha^{(d)} = L^{(d)} \times \alpha$
7 Generate $\theta^{(d)} = (\theta_{l_1}, \ldots, \theta_{l_{M_d}})^T \sim Dir(\cdot \mid \alpha^{(d)})$
8 For each i in $\{1, \ldots, N_d\}$:
9 Generate $z_i \in \{\lambda_1^{(d)}, \ldots, \lambda_{M_d}^{(d)}\} \sim Mult(\cdot \mid \theta^{(d)})$
10 Generate $w_i \in \{1, \ldots, V\} \sim Mult(\cdot \mid \beta_{z_i})$

Figure 2: L-LDA generative process (Ramage et al. 2009)

$$P(w|a) \approx P(w|d) = \sum_t P(w|t)P(t|d) \qquad (4)$$

3.3 Labeled LDA

L-LDA (Ramage et al., 2009) extends standard LDA to include supervision for specific target categories, yet in a different way: (i) The generative process includes a second observed variable, i.e. each document is explicitly labeled with a target category. A document may be labeled with an arbitrary number of categories; unlabeled documents are also possible. However, L-LDA permits only binary assignments of categories to documents; probabilistic weights over categories are not intended. (ii) Contrary to LDA, where the number of topics has to be specified in advance, L-LDA sets this parameter to the number of unique target categories. Moreover, the model is constrained such that documents may be assigned only those topics that correspond to their observable category label(s). That is, latent topics t in the standard formulation of LDA (2) are constrained to correspond to explicit labels a.

More specifically, L-LDA extends the generative process of LDA by constraining the topic distributions over documents $\theta^{(d)}$ to only those topics that correspond to the document's set of labels $\Lambda^{(d)}$. This is done by projecting the parameter vector of the Dirichlet topic prior α to a lower-dimensional vector $\alpha(d)$ whose topic dimensions correspond to the document labels.

This extension is integrated in steps 5 and 6 of Fig. 2: First, in step 5, the document's labels $\Lambda^{(d)}$ are generated for each topic k. The resulting vector of document's labels $\lambda^{(d)} = \{k \mid \Lambda_k^{(d)} = 1\}$ is used to define a document-specific label projection matrix

$L^{(d)}_{|\lambda^{(d)}|\times K}$, such that $L^{(d)}_{ij} = 1$ if $\lambda^{(d)}_i = j$, and 0 otherwise. This matrix is used in step 6 to project the Dirichlet topic prior α to a lower-dimensional vector $\alpha^{(d)}$, whose topic dimensions correspond to the document labels. Topic proportions are then, in step 7, generated for this reduced parameter space.

In our instantiation of L-LDA, we collect pseudo-documents for attributes exactly as for C-LDA. Documents are labeled with exactly one category, the attribute noun. Note that, even though the relationship between documents and topics is fixed, the one between topics and words is not. Any word occurring in more than one document will be assigned a non-zero probability for each corresponding topic.

Thus, with regard to attribute modeling, C-LDA and L-LDA build an interesting pair of opposites: The L-LDA model assumes that attributes are semantically primitive in the sense that they cannot be decomposed into smaller topical units, whereas words may be associated with several attributes at the same time. C-LDA, at the other end of the spectrum, licenses semantic variability on both the attribute and the word level. Particularly, a word might be associated with some of the topics underlying an attribute, but not with all of them, and an attribute can be characterized by multiple topics.

3.4 Vector Space Framework

For integrating the information obtained from C-LDA or L-LDA into a distributional VSM, we follow Hartung and Frank (2010): Adjectives and nouns are modeled as independent semantic vectors along their relationship to attributes; the most prominent attribute(s) that represent the hidden meaning of adjective-noun phrases are selected from their composition (cf. Fig. 1).

The dimensions of the VSM are set to the preselected attributes. Semantic vectors are computed for all adjectives and nouns occurring at least five times in the pseudo-documents. Vector component values $v_{\langle w,a\rangle}$ are derived from the C-LDA and L-LDA models in different ways: with C-LDA we obtain $P(w|a)$ by approximation from $P(w|d)$ (cf. equation (4)), while in L-LDA we obtain $P(w|a)$ directly from the induced topic-word distribution ϕ_t, through labeled topics $t = a$ (cf. equation (2)).

Vector composition is defined as *vector multiplication* ($\times$) or *vector addition* ($+$).

For attribute selection on the composed vector, we use two methods we found to perform best in Hartung and Frank (2010): Entropy Selection (ESel) and Most Prominent Component (MPC). ESel measures entropy over the vector components to identify components that encode a high amount of information. It selects all attributes that lead to an increase of entropy when suppressed from the vector representation. If no informative components can be detected in a vector due to a very broad, flat distribution of the probability mass (cf. $\vec{b}$ in Fig. 1), ESel yields an empty list. MPC always chooses exactly one vector component, i.e. the one with the highest value.

4 Experimental Settings

Attribute selection over small and large semantic spaces. We evaluate the performance of the VSMs based on C-LDA and L-LDA in two experimental settings, contrasting the problem of attribute selection on semantic spaces of radically different dimensionality, using sets of 10 vs. 206 attributes.

Evaluation measures. We evaluate against two gold standards consisting of adjective-noun phrases (or adjective-noun pairs) and their associated attribute meanings. We report precision, recall and f_1-score. Where appropriate, we test differences in the performance of various model configurations for statistical significance in a randomized permutation test (Yeh, 2000), using the `sigf` tool (Padó, 2006).

Baselines. We compare our models against two baselines, PATTVSM and DEPVSM. PATTSVM is reconstructed from Hartung and Frank (2010). It is grounded in a selection of lexical patterns that identify the target elements (adjectives and nouns) for the vector basis elements (i.e., the attribute nouns) in a local context window. The component values are defined using raw frequency counts over the extracted patterns. DEPVSM is similar to PATTVSM; however, it relies on dependency paths that connect the target elements and attributes in local contexts. The paths are identical to the ones used for constructing pseudo-documents in C-LDA and L-LDA. As in PATTVSM, the vector components are set to raw frequencies over extracted paths.

Implementations. To implement our models, we rely on MALLET (McCallum, 2002) for C-LDA and

the Stanford Topic Modeling Toolbox[3] for L-LDA. In both cases, we run 1000 iterations of Gibbs sampling, using default values for all hyperparameters.

Data set for attribute selection over 10 attributes. The first experiment is conducted on the data set used in Hartung and Frank (2010). It consists of 100 adjective-noun pairs manually annotated for ten attributes: COLOR, DIRECTION, DURATION, SHAPE, SIZE, SMELL, SPEED, TASTE, TEMPERATURE, WEIGHT. To enable comparison, the dimensions of our models are set to exactly these attributes.

Data set for attribute selection over a large semantic space (206 attributes). In the second experiment, we max out the attribute selection task to a much larger set of attributes in order to analyze the difficulty of the task on more representative data. We automatically construct a data set of adjective-noun phrases labeled with appropriate attributes from WordNet 3.0 (Fellbaum, 1998), relying on the assumption that examples given in glosses correspond to the respective word sense of the adjective. We first extract all adjectives that are linked to at least one attribute synset by the `attribute` relation. Next, we run the glosses of these adjectives (3592 in number) through TreeTagger (Schmid, 1994) to find examples of adjectives modifying nouns in attributive constructions. The resulting adjective-noun phrases are labeled with the attribute label linked to the given adjective sense.

This method yields 7901 labeled adjective-noun phrases. They are divided into development and test data according to a sampling procedure that respects the following criteria: (i) Both sets must contain all attributes with an equal number of phrases for each attribute; (ii) phrases with both elements contained in CoreWordNet[4] are preferred, while others are only considered if necessary to satisfy the first criterion. This procedure yields 496/345 phrases in the development/test set, distributed over 206 attributes[5].

[3] `http://nlp.stanford.edu/software/tmt/`.

[4] A subset of WordNet restricted to the 5000 most frequently used word senses. Available from: `http://wordnetcode.princeton.edu/standoff-files/core-wordnet.txt`

[5] If an attribute provides only one example, this was added to the development set. Therefore, the test set only comprises 206 attributes, while all models were trained on 262 attributes obtained from WordNet in the first extraction step.

Training data. The pseudo-documents are collected from dependency paths obtained from section 2 of the parsed pukWaC corpus (Baroni et al., 2009).

5 Discussion of Results

5.1 Experiment 1

In Experiment 1, we evaluate the performance of C-LDA and L-LDA on the attribute selection task over 10 attributes against the pattern-based and dependency-based models PATTVSM and DEPVSM as competitive baselines. Besides a comparison to standard VSMs, we are especially interested in the relative performance of the LDA models. Given that C-LDA and L-LDA estimate attribute-specific topic distributions in the structured pseudo-documents under different assumptions regarding the correspondence of attributes and topics (cf. Sec. 3.2 and 3.3), we expect the two LDA variants to differ in their capability to capture the topic distributions in the labeled pseudo-documents.

5.1.1 Attribute Selection for 10 Attributes

Tables 1 and 2 summarize the results for attribute selection over 10 attributes against the labeled adjective-noun pairs in the test set, using ESel and MPC as selection functions on vectors composed by multiplication (Table 1) and addition (Table 2). The results reported for C-LDA correspond to the best performing model (with number of topics set to 42, as this setting yields the best and most constant results over both composition operators).

C-LDA shows highest f-scores and recall over all settings, and highest precision with vector addition.[6] In line with Mitchell and Lapata (2010) (cf. Sec. 2), we obtain the best overall results with vector addition (ESel: P: 0.55, R: 0.66, F: 0.61; MPC: P: 0.59, R: 0.71, F: 0.64). The difference between C-LDA and L-LDA is small but significant for vector multiplication; for vector addition, it is not significant.

Compared to the LDA models, the VSM baselines

[6] In Tables 1 and 2, statistical significance of the differences between the models is marked by the superscripts L, D and P, denoting a significant difference over **L**-LDA, **D**epVSM and **P**attVSM, respectively. All differences reported are significant at $p < 0.05$, except for the difference between C-LDA and L-LDA in Table 3 ($p < 0.1$).

	ESel			MPC		
	P	R	F	P	R	F
C-LDA	0.58	**0.65**	**0.61**^{L,P}	0.57	**0.64**	**0.60**
L-LDA	**0.68**	0.54	0.60^{D}	0.55	0.61	0.58^{D}
DepVSM	0.48	0.58	0.53^{P}	0.57	0.60	0.58
PattVSM	0.63	0.46	0.54	**0.60**	0.58	0.59

Table 1: Attribute selection over 10 attributes ($\times$)

	ESel			MPC		
	P	R	F	P	R	F
C-LDA	**0.55**	**0.66**	**0.61**^{D,P}	**0.59**	**0.71**	**0.64**
L-LDA	0.53	0.57	0.55^{D,P}	0.50	0.45	0.47^{D,P}
DepVSM	0.38	0.65	0.48^{P}	0.57	0.60	0.58
PattVSM	0.71	0.35	0.47	0.47	0.56	0.51

Table 2: Attribute selection over 10 attributes ($+$)

are competitive, but tend to perform lower. This effect is statistically significant for ESel with vector multiplication: each of the LDA models statistically significantly outperforms one of the VSM models, DEPVSM and PATTVSM. With ESel and vector addition, both LDA models outperform both VSM models statistically significantly. The LDA$_{ESel,+}$ models outperform the PATTVSM$_{ESel,+}$ model of Hartung and Frank (2010) by a high margin in f-score: +0.14 for C-LDA; +0.08 for L-LDA. Compared to the stronger multiplicative settings PATTVSM$_{ESel,\times}$ and PATTVSM$_{MPC,\times}$ this still represents a plus of +0.07 and +0.02 in f-score, respectively. We further observe a clear improvement of the LDA models over the VSM models in terms of recall (+0.20, C-LDA$_{ESel,+}$ vs. PATTVSM$_{ESel,\times}$), at the expense of some loss in precision (-0.08, C-LDA$_{ESel,+}$ vs. PATTVSM$_{ESel,\times}$). This clearly confirms a stronger generalization power of LDA compared to VSM models.

With regard to selection functions, we observe that MPC tends to perform better for the VSM models, while ESel is more suitable in the LDA models.

Figures 3 and 4 display the overall performance curve ranging over different topic numbers for C-LDA$_{ESel,+}$ and C-LDA$_{ESel,\times}$ – compared to the remaining models that are not dependent on topic size. For topic numbers smaller than the attribute set size, C-LDA underperforms, for obvious reasons. Increasing ranges of topic numbers to 60 does not show a linear effect on performance. Parameter settings with performance drops below the VSM baselines are rare, which holds particularly for vector addition at topic ranges larger than 10. With vector addition, C-LDA outperforms L-LDA in almost all configurations, yet at an overall lower performance level of L-LDA (0.55 with addition vs. 0.6 with multiplication). Note that in the multiplicative setting, C-LDA reaches the performance of L-LDA only in its best configurations, while with vector addition it obtains high performance that exceeds L-LDA's top f-score of 0.6 for topic ranges between 10 and 20.

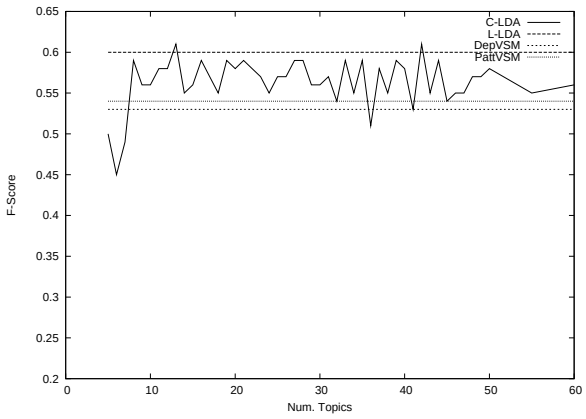


Figure 3: Performance of C-LDA$_{ESel,\times}$ for different topic numbers, compared against all other models

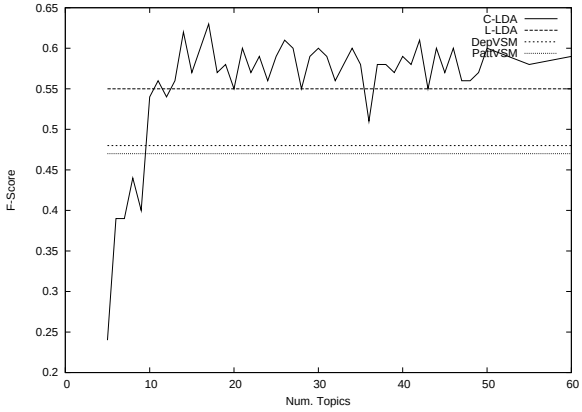


Figure 4: Performance of C-LDA$_{ESel,+}$ for different topic numbers, compared against all other models

Based on these observations, vector addition seems to offer the more robust setting for C-LDA, the model that is less strict with regard to topic-attribute correspondences. Vector multiplication, on the other hand, is more suitable for L-LDA and its stricter association of topics with class labels.

5.1.2 Smoothing Power of LDA Models

Our hypothesis was that LDA models should be better suited for dealing with sparse data, compared

	ESel			MPC		
	P	R	F	P	R	F
C-LDA	**0.39**	**0.31**	**0.35**	**0.37**	**0.27**	**0.32**
L-LDA	0.30	0.18	0.23	0.20	0.18	0.19
DepVSM	0.20	0.10	0.13	0.37	0.26	0.30
PattVSM	0.00	0.00	0.00	0.00	0.00	0.00

Table 3: Performance figures on sparse vectors ($\times$)

	ESel			MPC		
	P	R	F	P	R	F
C-LDA	**0.43**	**0.33**	**0.38**	**0.44**	**0.28**	**0.34**
L-LDA	0.34	0.16	0.22	0.37	0.18	0.24
DepVSM	0.16	0.17	0.17	0.36	0.21	0.27
PattVSM	0.13	0.04	0.06	0.17	0.25	0.20

Table 4: Performance figures on sparse vectors ($+$)

to pattern-based or purely distributional approaches. While this is broadly confirmed in the above results by global gains in recall, we conduct a special evaluation focused on those pairs in the test set that suffer from sparse data. We selected all adjective and noun vectors that did not yield any positive component values in the PATTVSM model. The 22 adjective-noun pairs in the test set affected by these 'zero vectors' were evaluated using the remaining models.

The results in Tables 3 and 4 yield a very clear picture: C-LDA obtains highest precision, recall and f-score across all settings, followed by L-LDA and DEPVSM$_{ESel}$, while their ranks are reversed when using MPC. Again, MPC works better for the VSM models, ESel for the LDA models. Vector addition performs best for C-LDA with f-scores of 0.38 and 0.34 – outperforming the pattern-based results on sparse vectors by orders of magnitude.

5.2 Experiment 2

Experiment 2 is designed to max out the space of attributes to be modeled, to assess the capacity of both LDA models and the DEPVSM baseline model in the attribute selection task on a large attribute space.[7] In contrast to Experiment 1, with its confined semantic space of 10 target attributes, this represents a huge undertaking.

5.2.1 Large-scale Attribute Selection

Table 5 (column **all**) displays the performance of all models on attribute selection over a range of 206 dimensions, contrasting vector addition and multiplication. The number of topics was set to 400. As the overall performance is close to 0 for both composition methods, no parameter setting can be identified as particularly suited for this large-scale attribute selection task. The differences between the three models are very small and not significant[8].

[7] We did not apply PATTVSM to this large-scale experiment, as only poor performance can be expected.

	all		property	
	$\times$	$+$	$\times$	$+$
C-LDA	0.04	0.02	$0.18^{L,D}$	0.10^{D}
L-LDA	0.03	0.04	0.15	0.15
DepVSM	0.02	0.02	0.12	0.07

Table 5: Performance figures (in f-score) of C-LDA$_{ESel}$ on 206 (all) and 73 property attributes (property)

	all			property		
	P	R	F	P	R	F
WIDTH	0.67	1.00	0.80	1.00	0.50	0.67
WEIGHT	0.80	0.57	0.67	0.50	0.57	0.53
MAGNETISM	0.50	1.00	0.67			
SPEED	0.50	0.50	0.50	1.00	0.50	0.67
TEXTURE	0.33	1.00	0.50	0.33	1.00	0.50
DURATION	0.50	0.50	0.50	1.00	1.00	1.00
TEMPERATURE	0.30	0.75	0.43	0.43	0.75	0.55
AGE	0.33	0.50	0.40			
THICKNESS	1.00	0.25	0.40	0.50	0.13	0.20
DEGREE	1.00	0.20	0.33			
LENGTH	0.17	1.00	0.29	0.50	1.00	0.67
DEPTH	1.00	0.14	0.25	1.00	0.86	0.92
ACTION	0.17	0.50	0.25			
LIGHT	0.33	0.17	0.22	0.20	0.17	0.18
POSITION	0.14	0.25	0.18	0.20	0.25	0.22
SHARPNESS				1.00	1.00	1.00
SERIOUSNESS				0.50	1.00	0.67
COLOR	0.13	0.25	0.17	0.29	0.50	0.36
LOYALTY				1.00	1.00	1.00
average	0.49	0.54	0.51	0.63	0.63	0.63

Table 6: Attribute selection on 206 attributes (all) and 73 property attributes (property); performance figures of C-LDA$_{ESel,\times}$ for best attributes (F>0)

5.2.2 Focused Evaluation and Data Analysis

To gain a deeper insight into the modeling capacity of the LDA models for this large-scale selection task, Table 6 (column **all**) presents a partial evaluation of attributes that could be assigned to adjective-noun pairs with an f-score >0 by C-LDA$_{ESel,\times}$.

Despite the disappointing overall performance of

[8] Again, statistically significant differences are marked by superscripts (cf. footnote 6). All differences reported are significant at $\alpha < 0.05$.

	prediction	correct
thin layer	THICKNESS	THICKNESS
heavy load	WEIGHT	WEIGHT
shallow water	DEPTH	DEPTH
short holiday	DURATION	DURATION
attractive force	MAGNETISM	MAGNETISM
short hair	LENGTH	LENGTH
serious book	DIFFICULTY	MIND
blue line	COLOR	UNION
weak president	POSITION	POWER
fluid society	REPUTE	CHANGEABLENESS
short flight	DISTANCE	DURATION
rough bark	TEXTURE	EVENNESS
faint heart	CONSTANCY	COWARDICE

Table 7: Sample of correct and false predictions of C-LDA$_{ESel,\times}$ in Experiment 2

the LDA models on this large attribute space, it is remarkable that C-LDA is able to induce distinctive topic distributions for a number of attributes with up to 0.51 f-score with balanced precision and recall, a moderate drop of only -0.10 relative to the corresponding model induced over 10 attributes.

Raising the attribute selection task from 10 to 206 attributes poses a true challenge to our models, by the sheer size and diversity of the semantic space considered. Table 7 gives an insight into the nature of the data and the difficulty of the task, by listing correct and false preditions of C-LDA for a small sample of adjective-noun pairs. Possible explanations for false predictions are manifold, among them near misses (e.g. *serious book*, *weak president*, *short flight*, *rough bark*), idiomatic expressions (e.g. *faint heart*, *blue line*) or questionable labels provided by WordNet (e.g. *serious book*).

As seen above, C-LDA achieves relatively high performance figures on selected attributes (cf. Table 6, col. **all**). In order to identify what makes these attributes different from others that resist successful modeling, we investigated three factors: (i) the amount of training data available for each attribute, (ii) the ambiguity rate per attribute, and (iii) their ontological subtype.

(i) Measuring the dependence between training data size and f-score per attribute shows that a large amount of training data is generally helpful, but not the decisive factor (Pearson's $r = 0.19$, $p < 0.01$).

(ii) The ambiguity rate AR_{attr} per attribute $attr$ is computed by averaging over all test pairs TP_{attr} labeled with $attr$, counting the total number of attributes $attr'$ that are associated with each adjective in pairs $\langle adj, n\rangle \in TP_{attr}$ in WordNet:

$$AR_{attr} = \frac{\sum_{attr'} \sum_{\langle adj,n\rangle \in TP_{attr}} |\langle adj, attr'\rangle_{WN}|}{|TP_{attr}|}$$

Correlating this figure with the performance per attribute in terms of f-score yields only a small positive correlation (Pearson's $r = 0.23$, $p < 0.01$). In fact, the qualitative analysis in Table 7 shows that C-LDA is capable of assigning meaningful attributes to adjective-noun phrases not only in easy, but also ambiguous cases (cf. *shallow water*, where DEPTH is the only attribute provided for *shallow* in WordNet vs. *short holiday*, *short hair* or *short flight*).

(iii) Although the 206 attributes used in Exp. 2 are rather diverse, including concepts such as HEIGHT, KINDNESS or INDIVIDUALITY, we observe a high number of attributes from Exp. 1 that are successfully modeled in Exp. 2 (5 out of 10, cf. column **all** in Table 6). Given that they are categorized into the *property* class in WordNet[9], we presume that the varying performance across attributes might be influenced by their ontological subtype. This hypothesis is validated in a replication of Exp. 2, with training data limited to the 73 attributes pertaining to the *property* subtype in WordNet. The test set was restricted accordingly, resulting in 112 pairs that are linked to a *property* attribute.

The overall performance of the models in this experiment is shown in Table 5 (column **property**): With vector multiplication, the best-performing operation across all models, all models benefit considerably (+0.10 or more). C-LDA shows the largest improvement, significantly outperforming both L-LDA and DEPVSM. With vector addition, the performance gains are slightly lower in general. In this setting, L-LDA shows higher f-score than C-LDA, though this difference is not statistically significant. Still, C-LDA significantly outranges DEPVSM. Note that we can not show a significant difference between C-LDA$_{ESel,\times}$ and L-LDA$_{ESel,+}$, so the comparison between these models remains inconclusive here. Note further that the affinity of C-LDA with vector addition and L-LDA with vector multiplication, respectively, is inverted in the large-scale experiment (cf. Table 5).

[9]WordNet separates attributes into *properties*, *qualities* and *states*, among several others.

While these overall results are far from satisfactory, they still clearly indicate that the LDA models work effectively for at least a subset of attributes, and outperform the VSM baseline.

Again, a more detailed analysis is given in Table 6 (column **property**), showing the performance of the best individual property attributes ($F>0$) in the restricted experiment. Average performance of the best property attributes with $F>0$, individually, amounts to $F=0.63$[10]. In comparison to the unrestricted setting (cf. column **all**), nearly all property attributes benefit from model training on selective data. Exceptions are WIDTH, WEIGHT, THICKNESS, AGE, DEGREE and LIGHT. Thus, apparently, some of the adjectives associated with non-property attributes in the full set provide some discriminative power that is helpful to distinguish property types.

In a qualitative analysis of the 133 non-property attributes filtered out in this experiment, we find that the WordNet-SUMO mapping (Niles, 2003) does not provide differentiating definitions for about 60% of these attributes, linking them instead to a single *subjective assessment attribute*. This suggests that in many cases the distinctions drawn by WordNet are too subtle even for humans to reproduce.

6 Conclusion

This paper explored the use of LDA topic models in a semantic labeling task that predicts attributes as 'hidden' meanings in the compositional semantics of adjective-noun phrases. LDA topic models are expected to alleviate sparsity problems of distributional VSMs as encountered in prior work, by incorporating latent semantic information about attribute nouns. We investigated two variants of LDA that employ different degrees of supervision for associating topics with attributes.

Our contributions are as follows. We proposed two LDA models for the attribute selection task that import supervision for a target category parameter in different ways: L-LDA (Ramage et al., 2009) embeds the target categories into the LDA process, by defining a 1:1 correspondence of topics and target categories. C-LDA, by contrast, does not affect the LDA generative process. Here, we heuristically equate pseudo-documents with target categories, to approximate category-specific word-topic distributions. By adhering to standard LDA, C-LDA accommodates a greater variety in the distributions of topics to attribute-specific documents and words, as compared to L-LDA. Combining standard LDA topic modeling with a means of interpreting the induced topics relative to a set of external categories, C-LDA offers greater flexibility and expressiveness.

Our experimental results show that modeling attributes as latent or explicit topics with C-LDA and L-LDA, respectively, outperforms the purely distributional baseline model DEPVSM and PATTVSM of prior work. Targeted evaluation on sparse data points confirms that LDA models help to overcome inherent sparsity effects of VSMs. C-LDA and L-LDA are close in performance in Experiment 1. C-LDA outperforms L-LDA only with optimal topic parameter settings.

Finally, we probed the modeling capacity of LDA and VSM models on a vast space of 206 attributes. This task proved to be extremely difficult. However, we obtain respectable results on a subset of attributes denoting properties, where C-LDA performs best in quantitative performance measures. It yields highest f-scores in full and partial evaluation – both with the full-size attribute model, and when training and testing is restricted to property attributes. The differences are small, but statistically significant between the LDA models and the VSM baseline in a setting restricted to property attributes.

Data analysis indicates that our models perform more robustly on concrete attributes in contrast to abstract attribute types that lack clear categorization. This suggests that our approach to attribute selection is most appropriate for detecting attributes that reflect clear ontological distinctions.

However, there is ample space for improvement. In Hartung and Frank (2011), we show that the quality of the noun vectors lags behind the adjective vectors. This clearly affects the performance of our models in cases where the semantic contribution of the noun is decisive for disambiguation. Future work will focus on ways to enhance the noun vector representations through additional contextual features, to make them denser and more articulated in structure.

[10] In comparison, L-LDA$_{ESel,\times}$ yields an average f-score of 0.47 for attributes with $F>0$ in the property setting.

References

Abdulrahman Almuhareb. 2006. *Attributes in Lexical Acquisition*. Ph.D. Dissertation, Department of Computer Science, University of Essex.

Marco Baroni and Alessandro Lenci. 2010. Distributional Memory. A General Framework for Corpus-based Semantics. *Computational Linguistics*, 36:673–721.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, East Stroudsburg, PA, pages 1183–1193.

M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation*, 43:209–226.

D. Blei and J. McAuliffe. 2007. Supervised topic models. *Neural Information Processing Systems*, 21.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *JMLR*, 3:993–1022.

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

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, Stroudsburg, PA. Association for Computational Linguistics.

Zellig Harris. 1968. *Mathematical Structures of Language*. Wiley.

Matthias Hartung and Anette Frank. 2010. A Structured Vector Space Model for Hidden Attribute Meaning in Adjective-Noun Phrases. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, Beijing, China, August.

Matthias Hartung and Anette Frank. 2011. Assessing interpretable, attribute-related meaning representations for adjective-noun phrases in a similarity prediction task. In *Proceedings of GEometrical Models of Natural Language Semantics (GEMS-2011)*, Edinburgh, UK.

Simon Lacoste-Julien, Fei Sha, and Michael I. Jordan. 2008. DiscLDA: Discriminative Learning for Dimensionality Reduction and Classification. In *NIPS*, volume 22.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1138–1147, Uppsala, Sweden.

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

Jeff Mitchell and Mirella Lapata. 2008. Vector-based Models of Semantic Composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June.

Jeff Mitchell and Mirella Lapata. 2009. Language Models Based on Semantic Composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, August 2009, pages 430–439, Singapore, August.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34:1388–1429.

Ian Niles. 2003. Mapping WordNet to the SUMO Ontology. In *Proceedings of the IEEE International Knowledge Engineering conference*, pages 23–26, June.

Diarmuid Ó Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444, Uppsala, Sweden, July. Association for Computational Linguistics.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

D. Prescher, S. Riezler, and M. Rooth. 2000. Using a probabilistic class-based lexicon for lexical ambiguity resolution. In *Proceedings of the 18th COLING*, pages 649–655.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, August 2009, pages 248–256.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A Latent Dirichlet Allocation Method for Selectional Preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Uppsala, Sweden, July. Association for Computational Linguistics.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 907–916. Association for Computational Linguistics, July.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing (NeMLaP). Manchester, U.K., 14–16 September 1994*.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the Fourth Conference on Computational Language Learning (CoNLL-2000) and the Second Learning Language in Logic Workshop*, Lisbon, Portugal, pages 947–953.

Semantic Topic Models: Combining Word Distributional Statistics and Dictionary Definitions

Weiwei Guo
Department of Computer Science,
Columbia University,
weiwei@cs.columbia.edu

Mona Diab
Center for Computational Learning Systems,
Columbia University,
mdiab@ccls.columbia.edu

Abstract

In this paper, we propose a novel topic model based on incorporating dictionary definitions. Traditional topic models treat words as surface strings without assuming predefined knowledge about word meaning. They infer topics only by observing surface word co-occurrence. However, the co-occurred words may not be semantically related in a manner that is relevant for topic coherence. Exploiting dictionary definitions explicitly in our model yields a better understanding of word semantics leading to better text modeling. We exploit WordNet as a lexical resource for sense definitions. We show that explicitly modeling word definitions helps improve performance significantly over the baseline for a text categorization task.

1 Introduction

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) serves as a data-driven framework in modeling text corpora. The statistical model allows variable extensions to integrate linguistic features such as syntax (Griffiths et al., 2005), and has been applied in many areas.

In LDA, there are two factors which determine the topic of a word: the topic distribution of the document, and the probability of a topic to emit this word. This information is learned in an unsupervised manner to maximize the likelihood of the corpus. However, this data-driven approach has some limitations. If a word is not observed frequently enough in the corpus, then it is likely to be assigned the dominant topic in this document. For example, the word *grease (a thick fatty oil)* in a political domain document should be assigned the topic *chemicals*. However, since it is an infrequent word, LDA cannot learn its correct semantics from the observed distribution, the LDA model will assign it the dominant document topic *politics*. If we look up the semantics of the word *grease* in a dictionary, we will not find any of its meanings indicating the *politics* topic, yet there is ample evidence for the *chemical* topic. Accordingly, we hypothesize that if we know the semantics of words in advance, we can get a better indication of their topics. Therefore, in this paper, we test our hypothesis by exploring the integration of word semantics explicitly in the topic modeling framework.

In order to incorporate word semantics from dictionaries, we recognize the need to model sense-topic distribution rather than word-topic distribution, since dictionaries are constructed at the sense level. We use WordNet (Fellbaum, 1998) as our lexical resource of choice. The notion of a sense in WordNet goes beyond a typical word sense in a traditional dictionary since a WordNet sense links senses of different words that have similar meanings. Accordingly, the sense for the first verbal entry for *buy* and for *purchase* will have the same sense id (and same definition) in WordNet, while they could have different meaning definitions in a traditional dictionary such as the Merriam Webster Dictionary or LDOCE. In our model, a topic will first emit a WordNet sense, then the sense will generate a word. This is inspired by the intuition that words are instantiations of concepts.

The paper is organized as follows: In Sections 2 and 3, we describe our models based on WordNet. In Section 4, experiment results on text categorization are presented. Moreover, we analyze both qualitatively and quantitatively the contribution of modeling definitions (by teasing out the contribution of explicit sense modeling in a word sense disambiguation task). Related work is introduced in Section 5. We conclude in Section 6 by discussing some possible future directions.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 552–561,
Edinburgh, Scotland, UK, July 27–31, 2011.

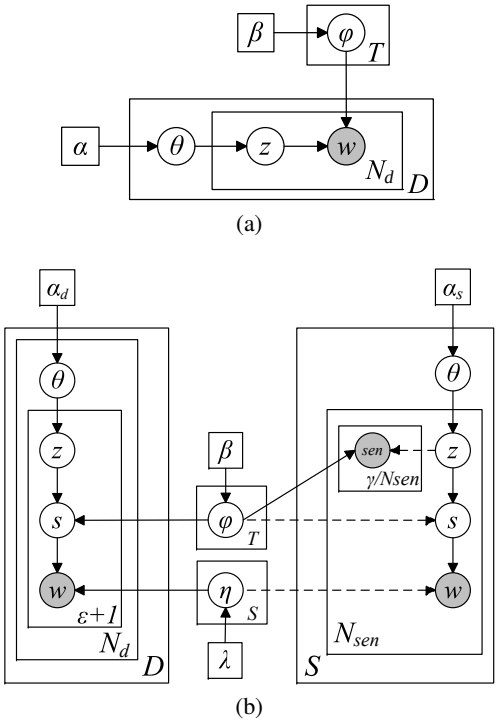


Figure 1: (a) LDA: Latent Dirichlet Allocation (Blei et al., 2003). (b) STM: Semantic topic model. The dashed arrows indicate the distributions (ϕ and η) and nodes (z) are not influenced by the values of pointed nodes.

2 Semantic Topic Model

2.1 Latent Dirichlet Allocation

We briefly introduce LDA where Collapsed Gibbs Sampling (Griffiths and Steyvers, 2004) is used for inference. In figure 1a, given a corpus with D documents, LDA will summarize each document as a normalized T-dimension topic mixture θ. Topic mixture θ is drawn from a Dirichlet distribution $Dir(\alpha)$ with a symmetric prior α. ϕ contains T multinomial distribution, each representing the probability of a topic z generating word w $p(w|z)$. ϕ is drawn from a Dirichlet distribution $Dir(\beta)$ with prior β.

In Collapsed Gibbs Sampling, the distribution of a topic for the word $w_i = w$ based on values of other data is computed as:

$$P(z_i = z|\mathbf{z_{-i}}, \mathbf{w}) \propto \frac{n^{(d)}_{-i,z} + \alpha}{n^{(d)}_{-i} + T\alpha} \times \frac{n^{w}_{-i,z} + \beta}{n_{-i,z} + W\beta} \quad (1)$$

In this equation, $n^{(d)}_{-i,z}$ is a count of how many words are assigned topic z in document d, excluding the topic of the ith word; $n^{w}_{-i,z}$ is a count of how many words $= w$ are assigned topic z, also excluding the topic of the ith word. Hence, the first fraction is the proportion of the topic in this document $p(z|\theta)$. The second fraction is the probability of topic z emitting word w. After the topics become stable, all the topics in a document construct the topic mixture θ.

2.2 Applying Word Sense Disambiguation Techniques

We add a sense node between the topic node and the word node based on two linguistic observations: a) **Polysemy**: many words have more than one meaning. A topic is more directly relevant to a word meaning (sense) than to a word due to polysemy; b) **Synonymy**: different words may share the same sense. WordNet explicitly models synonymy by linking synonyms to the same sense. In WordNet, each sense has an associated definition.

It is worth noting that we model the sense-word relation differently from (Boyd-Graber and Blei, 2007), where in their model words are generated from topics, then senses are generated from words. In our model, we assume that during the generative process, the author picks a concept relevant to the topic, then thinks of a best word that represents that concept. Hence the word choice is dependent on the relatedness of the sense and its fit to the document context.

In standard topic models, the topic of a word is sampled from the document level topic mixture θ. The underlying assumption is that all words in a document constitute the context of the target word. However, it is not the case in real world corpora. Titov and McDonald (2008) find that using global topic mixtures can only extract global topics in online reviews (e.g., Creative Labs MP3 players and iPods) and ignores local topics (product features such as portability and battery). They design the Multi-grain LDA where the local topic of a word is only determined by topics of surrounding sentences. In word sense disambiguation (WSD), an even narrower context is taken into consideration, for instance in graph based WSD models (Mihalcea, 2005), the choice of a sense for a word only depends on a local window whose size equals the length of the sentence. Later in (Sinha and Mihalcea, 2007; Guo and Diab, 2010; Li et al., 2010), people use a fixed window size containing around 12 neighbor words for WSD.

Accordingly, we adopt the WSD inspired local window strategy in our model. However, we do

not employ the complicated schema in (Titov and McDonald, 2008). We simply hypothesize that the surrounding ϵ words are semantically related to the considered word, and they construct a local sliding window for that target word. For a document d with N_d words, we represent it as N_d local windows – a window is created for each word. The model is illustrated in the left rectangle in figure 1b. The window size is fixed for each word: it contains $\epsilon/2$ preceding words, and $\epsilon/2$ following words. Therefore, a word in the original document will have ϵ copies, existing in $\epsilon+1$ local windows. Similarly, there are $\epsilon + 1$ pairs of topics/senses assigned for each word in the original document. Each window has a distribution θ_i over topics. θ_i will emit the topics of words in the window.

This approach enables us to exploit different context sizes without restricting it to the sentence length, and hence spread topic information across sentence boundaries.

2.3 Integrating Definitions

Intuitively, a sense definition reveals some prior knowledge on the topic domain: the definition of sense [*crime, offense, offence*] indicates a *legal* topic; the definition of sense [*basketball*] indicates a *sports* topic, etc. Therefore, during inference, we want to choose a topic/sense pair for each word, such that the topic is supported by the context θ and the sense definition also matches that topic.

Given that words used in the sense definitions are strongly relevant to the sense/concept, we set out to find the topics of those definition words, and accordingly assign the sense sen itself these topics. We treat a sense definition as a document and perform Gibbs sampling on it. We normalize definition length by a variable γ. Therefore, before the topic model sees the actual documents, each sense s has been sampled γ times. The γ topics are then used as a "training set", so that given a sense, ϕ has some prior knowledge of which topic it should be sampled from.

Consider the sense [*party, political party*] with a definition "an organization to gain political power" of length 6 when $\gamma = 12$. If topic model assigns *politics* topic to the words "organization political power", then sense [*party, political party*] will be sampled from $politics$ topic for $3 * \gamma/definitionLength = 6$ times.

We refer to the proposed model as Semantic Topic Model (figure 1b). For each window v_i in the document set, the model will generate a distribution of topics θ_i. It will emit the topics of $\epsilon + 1$ words in the window. For a word w_{ij} in window v_i, a sense s_{ij} is drawn from the topic, and then s_{ij} generates the word w_i. Sense-topic distribution ϕ contains T multinomial distributions over all possible senses in the corpus drawn from a symmetric Dirichlet distribution $Dir(\beta)$. From WordNet we know the set of words $W(s)$ that have a sense s as an entry. A sense s can only emit words from $W(s)$. Hence, for each sense s, there is a multinomial distribution η_s over $W(s)$. All η are drawn from symmetric $Dir(\lambda)$.

On the definition side, we use a different prior α_s to generate a topic mixture θ. Aside from generating s_i, z_i will deterministically generate the current sense sen for γ/N_{sen} times (N_{sen} is the number of words in the definition of sense sen), so that sen is sampled γ times in total.

The formal procedure of generative process is the following:

For the definition of sense sen:
- choose topic mixture $\theta \sim Dir(\alpha_s)$.
- for each word w_i:
  - choose topic $z_i \sim Mult(\theta)$.
  - choose sense $s_i \sim Mult(\phi_{z_i})$.
  - deterministically choose sense $sen \sim Mult(\phi_{z_i})$ for γ/N_{sen} times.
  - choose word $w_i \sim Mult(\eta_{s_i})$.

For each window v_i in a document:
- choose local topic mixture $\theta_i \sim Dir(\alpha_d)$.
- for each word w_{ij} in v_i:
  - choose topic $z_{ij} \sim Mult(\theta_i)$.
  - choose sense $s_{ij} \sim Mult(\phi_{z_{ij}})$.
  - choose word $w_{ij} \sim Mult(\eta_{s_{ij}})$.

2.4 Using WordNet

Since definitions and documents are in different genre/domains, they have different distributions on senses and words. Besides, the definition sets contain topics from all kinds of domains, many of which are irrelevant to the document set. Hence we prefer ϕ and η that are specific for the document set, and we do not want them to be "corrupted" by the text in the definition set. Therefore, as in figure 1b, the dashed lines indicate that when we estimate ϕ and η, the topic/sense pair and sense/word pairs in the definition set are not considered.

WordNet senses are connected by relations such as synonymy, hypernymy, similar attributes, etc.

We observe that neighboring sense definitions are usually similar and are in the same topic domain. Hence, we represent the definition of a sense as the union of itself with its neighboring sense definitions pertaining to WordNet relations. In this way, the definition gets richer as it considers more data for discovering reliable topics.

3 Inference

We still use Collapsed Gibbs Sampling to find latent variables. Gibbs Sampling will initialize all hidden variables randomly. In each iteration, hidden variables are sequentially sampled from the distribution conditioned on all the other variables. In order to compute the conditional probability $P(z_i = z, s_i = s|\mathbf{z_{-i}}, \mathbf{s_{-i}}, \mathbf{w})$ for a topic/sense pair, we start by computing the joint probability $P(\mathbf{z}, \mathbf{s}, \mathbf{w}) = P(\mathbf{z})P(\mathbf{s}|\mathbf{z})P(\mathbf{w}|\mathbf{s})$. Since the generative processes are not exactly the same for definitions and documents, we need to compute the joint probability differently. We use a type specific subscript to distinguish them: $P_s(\cdot)$ for sense definitions and $P_d(\cdot)$ for documents.

Let sen be a sense. Integrating out θ we have:

$$P_s(\mathbf{z}) = \left(\frac{\Gamma(T\alpha_s)}{\Gamma(\alpha_s)^T}\right)^S \prod_{sen=1}^{S} \frac{\prod_z \Gamma(n_z^{(sen)} + \alpha_s)}{\Gamma(n^{(sen)} + T\alpha)} \tag{2}$$

where $n_z^{(sen)}$ means the number of times a word in the definition of sen is assigned to topic z, and $n^{(sen)}$ is the length of the definition. S is all the potential senses in the documents.

We have the same formula of $P(\mathbf{s}|\mathbf{z})$ and $P(\mathbf{w}|\mathbf{s})$ for definitions and documents. Similarly, let n_z be the number of words in the documents assigned to topic z, and n_z^s be the number of times sense s assigned to topic z. Note that when s appears in the superscript surrounded by brackets such as $n_z^{(s)}$, it denotes the number of words assigned to topics z in the definition of sense s. By integrating out ϕ we obtain the second term:

$$P(\mathbf{s}|\mathbf{z}) = \left(\frac{\Gamma(S\beta)}{\Gamma(\beta)^S}\right)^T \prod_{z=1}^{T} \frac{\prod_s \Gamma(n_z^s + n_z^{(s)}\gamma/n^{(s)} + \beta)}{\Gamma(n_z + \sum_{s'} n_z^{(s')}\gamma/n^{(s')} + S\beta)} \tag{3}$$

At last, assume n_s denotes the number of sense s in the documents, and n_s^w denotes the number of sense s to generate the word w, then integrating out η we have:

$$P(\mathbf{w}|\mathbf{s}) = \prod_{s=1}^{S} \frac{\Gamma(|W(s)|\lambda)}{\Gamma(\lambda)^{|W(s)|}} \frac{\prod_w^{W(s)} \Gamma(n_s^w + \lambda)}{\Gamma(n_s + |W(s)|\lambda)}$$

With equation 2-4, we can compute the conditional probability $P_s(z_i = z, s_i = s|\mathbf{z_{-i}}, \mathbf{s_{-i}}, \mathbf{w})$ for a sense-topic pair in the sense definition. Let sen_i be the sense definition containing word w_i, then we have:

$$P_s(z_i = z, s_i = s|\mathbf{z_{-i}}, \mathbf{s_{-i}}, \mathbf{w}) \propto \frac{n_{-i,z}^{(sen_i)} + \alpha_s}{n_{-i}^{(sen_i)} + T\alpha_s} \frac{n_z^s + n_{-i,z}^{(s')}\gamma/n^{(s')} + \beta}{n_z + \sum_{s'} n_{-i,z}^{(s')}\gamma/n^{(s')} + S\beta} \frac{n_s^w + \lambda}{n_s + |W(s)|\lambda} \tag{5}$$

The subscript $-i$ in expression n_{-i} denotes the number of certain events excluding word w_i. Hence the three fractions in equation 5 correspond to the probability of choosing z from θ_{sen}, choosing s from z and choosing w from s. Also note that our model defines s that can only generate words in $W(s)$, therefore for any word $w \notin W(s)$, the third fraction will yield a 0.

The probability for documents is similar to that for definitions except that there is a topic mixture for each word, which is estimated by the topics in the window. Hence $P_d(\mathbf{z})$ is estimated as:

$$P_d(\mathbf{z}) = \prod_i \frac{\Gamma(T\alpha_d)}{\Gamma(\alpha_d)^T} \frac{\prod_z \Gamma(n_z^{(v_i)} + \alpha_d)}{\Gamma(n^{(v_i)} + T\alpha_d)} \tag{6}$$

Thus, the conditional probability for documents can be estimated by cancellation terms in equation 6, 3, and 4:

$$P_d(z_{ij} = z, s_{ij} = s|\mathbf{z_{-ij}}, \mathbf{s_{-ij}}, \mathbf{w}) \propto \frac{n_{-ij,z}^{(v_i)} + \alpha_d}{n_{-ij}^{(v_i)} + T\alpha_d} \frac{n_{-ij,z}^s + n_z^{(s')}\gamma/n^{(s')} + \beta}{n_{-ij,z} + \sum_{s'} n_z^{(s')}\gamma/n^{(s')} + S\beta} \frac{n_{-ij,s}^w + \lambda}{n_{-ij,s} + |W(s)|\lambda} \tag{7}$$

3.1 Approximation

In current model, each word appears in $\epsilon + 1$ windows, and will be generated $\epsilon + 1$ times, so there will be $\epsilon + 1$ pairs of topics/senses sampled for each word, which requires a lot of additional computation (proportional to context size ϵ). On the other hand, it can be imagined that the set of values $\{z_{ij}, s_{ij}|j - \epsilon/2 \leq i \leq j + \epsilon/2\}$ in different windows v_i should roughly be the same, since they are hidden values for the same word w_j. Therefore, to reduce computation complexity during Gibbs sampling, we approximate the values of $\{z_{ij}, s_{ij}|\ i \neq j\}$ by the topic/sense (z_{jj}, s_{jj}) that are generated from window v_j. That is, in Gibbs sampling, the algorithm does not actually sample the values of $\{z_{ij}, s_{ij}, |\ i \neq j\}$; instead, it directly assumes the sampled values are z_{jj}, s_{jj}.

4 Experiments and Analysis

Data: We experiment with several datasets, namely, the Brown Corpus (Brown), New York Times (NYT) from the American National Corpus, Reuters (R20) and WordNet definitions. In a preprocessing step, we remove all the non-content words whose part of speech tags are not one of the following set {*noun, adjective, adverb, verb*}. Moreover, words that do not have a valid lemma in WordNet are removed. For WordNet definitions, we remove stop words hence focusing on relevant content words.

Corpora statistics after each step of preprocessing is presented in Table 1. The column *WN token* lists the number of word#pos tokens after preprocessing. Note that now we treat word#pos as a word token. The column *word types* shows corresponding word#pos types, and the total number of possible sense types is listed in column *sense types*. The DOCs size for WordNet is the total number of senses defined in WordNet.

Experiments: We design two tasks to test our models: (1) text categorization task for evaluating the quality of values of topic nodes, and (2) a WSD task for evaluating the quality of the values of the sense nodes, mainly as a diagnostic tool targeting the specific aspect of sense definitions incorporation and distinguish that component's contribution to text categorization performance. We compare the performance of four topic models. (a) LDA: the traditional topic model proposed in (Blei et al., 2003) except that it uses Gibbs Sampling for inference. (b) LDA+def: is LDA with sense definitions. However they are not explicitly modeled; rather they are treated as documents and used as augmented data. (c) STM0: the topic model with an additional explicit sense node in the model, but we do not model the sense definitions. And finally (d) STMn is the full model with definitions explicitly modeled. In this setting n is the γ value. We experiment with different γ values in the STM models, and investigate the semantic scope of words/senses by choosing different window size ϵ. We report mean and standard deviation based on 10 runs.

It is worth noting that a larger window size ϵ suggests documents have larger impact on the model (ϕ, η) than definitions, since each document word has ϵ copies. This is not a desirable property when we want to investigate the weight of definitions by choosing different γ values. Accordingly, we only use z_{jj}, s_{jj}, w_{jj} to estimate ϕ, η, so that the impact of documents is fixed. This makes more sense, in that after the approximation in section 3.1, there is no need to use $\{z_{ij}, s_{ij}, \mid i \neq j\}$ (they have the same values as z_{jj}, s_{jj}).

4.1 Text Categorization

We believe our model can generate more "correct" topics by looking into dictionaries. In topic models, each word is generalized as a topic and each document is summarized as the topic mixture θ, hence it is natural to evaluate the quality of inferred topics in a text categorization task. We follow the classification framework in (Griffiths et al., 2005): first run topic models on each dataset **individually** without knowing label information to achieve document level topic mixtures, then we employ Naive Bayes and SVM (both implemented in the WEKA Toolkit (Hall et al., 2009)) to perform classification on the topic mixtures. For all document, the features are the percentage of topics. Similar to (Griffiths et al., 2005), we assess inferred topics by the classification accuracy of 10-fold cross validation on each dataset.

We evaluate our models on three datasets in the cross validation manner: The Brown corpus which comprises 500 documents grouped into 15 categories (same set used in (Griffiths et al., 2005)); NYT comprising 800 documents grouped into the 16 most frequent label categories; Reuters (R20) comprising 8600 documents labeled with the most frequent 20 categories. In R20, combination of categories is treated as separate category labels, so $money, interest$ and $interest$ are considered different labels.

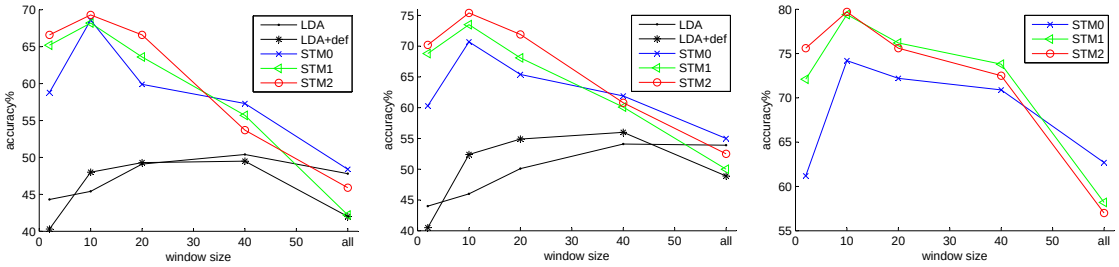
For the three datasets, we use the Brown corpus only as a tuning set to decide on the topic model parameters for all of our experimentation, and use the optimized parameters directly on NYT and R20 without further optimization.

4.1.1 Classification Results

Searching γ and ϵ on Brown: The classification accuracy on the Brown corpus with different ϵ and γ values using Naive Bayes and SVM are presented in figure 2a and 2b. In this section, the number of topics T is set to 50. The possible ϵ values in the horizontal axis are *2, 10, 20, 40, all*. The possible γ values are *0, 1, 2*. Note that $\epsilon = all$ means that no local window is used, and $\gamma = 0$ means definitions are not used. The hyper-

Corpus	DOCs size	orig tokens	content tokens	WN tokens	word types	sense types
Brown	500	1022393	580882	547887	27438	46645
NYT	800	743665	436988	393120	19025	37631
R20	8595	901691	450935	417331	9930	24834
SemCor	352	676546	404460	352563	28925	45973
WordNet	117659	1447779	886923	786679	42080	60567

Table 1: Corpus statistics



(a) Naive Bayes on Brown (b) SVM on Brown (c) SVM on NYT

Figure 2: Classification accuracy at different parameter settings

parameters are tuned as $\alpha_d = 0.1, \alpha_s = 0.01, \beta = 0.01, \lambda = 0.1$.

From figure 2, we observe that results using SVM have the same trend as Naive Bayes except that the accuracies are roughly 5% higher for SVM classifier. The results of LDA and LDA+def suggest that simply treating definitions as documents in an augmented data manner does not help. Comparing SMT0 with LDA in the same ϵ values, we find that explicitly modeling the sense node in the model greatly improves the classification results. The reason may be that words in LDA are independent isolated strings, while in STM0 they are connected by senses.

STM2 prefers smaller window sizes (ϵ less than 40). That means two words with a distance larger than 40 are not necessarily semantically related or share the same topic. This ϵ number also correlates with the optimal context window size of 12 reported in WSD tasks (Sinha and Mihalcea, 2007; Guo and Diab, 2010).

Classification results: Table 2 shows the results of our models using best tuned parameters of $\epsilon = 10, \gamma = 2$ on 3 datasets. We present three baselines in Table 2: (1) WEKA uses WEKA's classifiers directly on bag-of-words without topic modeling. The values of features are simply term frequency. (2) WEKA+FS performs feature selection using information gain before applying classification. (3) LDA, is the traditional topic model. Note that Griffiths et al.'s (2005) implementation of LDA achieve 51% on Brown corpus using Naive Bayes . Finally the Table illustrates the results obtained using our proposed models STM0 (γ=0) and STM2 (γ = 2).

It is worth noting that R20 (compared to NYT) is a harder condition for topic models. This is because fewer words (10000 distinct words versus 19000 in NYT) are frequently used in a large training set (8600 documents versus 800 in NYT), making the surface word feature space no longer as sparse as in the NYT or Brown corpus, which implies simply using surface words without considering the words distributional statistics – topic modeling – is good enough for classification. In (Blei et al., 2003) figure 10b they also show worse text categorization results over the SVM baseline when more than 15% of the training labels of Reuters are available for the SVM classifiers, indicating that LDA is less necessary with large training data. In our investigation, we report results on SVM classifiers trained on the whole Reuters training set. In our experiments, LDA fails to correctly classify nearly 10% of the Reuters documents compared to the WEKA baseline, however STM2 can still achieve significantly better accuracy (+4%) in the SVM classification condition.

Table 2 illustrates that despite the difference between NYT, Reuters and Brown (data size, genre, domains, category labels), exploiting WSD techniques (namely using a local window size coupled with explicitly modeling a sense node) yields

	Brown		NYT		R20	
	NB	SVM	NB	SVM	NB	SVM
WEKA	48	47.8	57	54.1	72.4	82.9
WEKA+FS	50	47.2	56.9	55.1	72.9	83.4
LDA	47.8±4.3	53.9±3.8	48.5±5.5	53.8±3.5	61.0±3.3	72.5±2.5
STM0	68.6±3.5	70.7±3.9	66.7±3.8	74.2±4.0	72.7±3.5	85.2±0.9
STM2	69.3±3.3	75.4±3.7	74.6±3.3	79.3±2.5	73±3.7	86.9±1.2

Table 2: Classification results on 3 datasets using hyperparameters tuned on Brown.

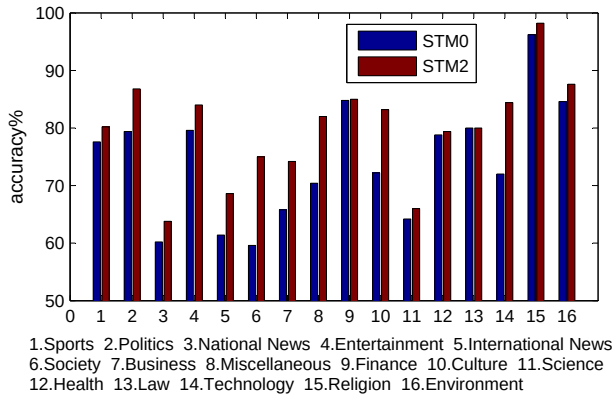


Figure 3: SVM accuracy on each category of NYT

significantly better results than all three baselines including LDA. Furthermore, explicit definition modeling as used in STM2 yields the best performance consistently overall.

Finally, in Figure 2c we show the SVM classification results on NYT in different parameter settings. We find that the NYT classification accuracy trend is consistent with that on the Brown corpus for each parameter setting of $\epsilon \in \{2, 10, 20, 40, all\}$ and $\gamma \in \{0, 1, 2\}$. This further proves the robustness of STMn.

4.2 Analysis on the Impact of Modeling Definitions

4.2.1 Qualitative Analysis

To understand why definitions are helpful in text categorization, we analyze the SVM performance of STM0 and STM2 ($\epsilon = 10$) on each category of NYT dataset (figure 3). We find STM2 outperforms STM0 in all categories. However, the largest gain is observed in *Society*, *Miscellaneous*, *Culture*, *Technology*. For *Technology*, we should credit WordNet definitions, since *Technology* may contain many infrequent technical terms, and STM0 cannot generalize the meaning of words only by distributional information due to their low frequency usage. However in some other domains, fewer specialized words are repeatedly used, hence STM0 can do as well as STM2.

For the other 3 categories, we hypothesize that these documents are likely to be a mixture of multiple topics. For example, a *Culture* news could contain topics pertaining to *religion, history, art*; while a *Society* news about crime could relate to *law, family, economics*. In this case, it is very important to sample a true topic for each word, so that ML algorithms can distinguish the *Culture* documents from the *Religion* ones by the proportion of topics. Accordingly, adding definitions should be very helpful, since it specifically defines the topic of a sense, and shields it from the influence of other "incorrect/irrelevant" topics.

4.2.2 Quantitative Analysis with Word Sense Disambiguation

A side effect of our model is that it sense disambiguates all words. As a means of analyzing and gaining some insight into the exact contribution of explicitly incorporating sense definitions (STMn) versus simply a sense node (STM0) in the model, we investigate the quality of the sense assignments in our models. We believe that the choice of the correct sense is directly correlated with the choice of a correct topic in our framework. Accordingly, a relative improvement of STMn over STM0 (where the only difference is the explicit sense definition modeling) in WSD task is an indicator of the impact of using sense definitions in the text categorization task.

WSD Data: We choose the all-words WSD task in which an unsupervised WSD system is required to disambiguate all the content words in documents. Our models are evaluated against the SemCor dataset. We prefer SemCor to all-words datasets available in Senseval-3 (Snyder and Palmer, 2004) or SemEval-2007 (Pradhan et al., 2007), since it includes many more documents than either set (350 versus 3) and therefore allowing more reliable results. Moreover, SemCor is also the dataset used in (Boyd-Graber et al., 2007), where a WordNet based topic model for WSD is introduced. The

	Total	Noun	Adjective	Adverb	Verb
sense annotated words	225992	86996	31729	18947	88320
polysemous words	187871	70529	21989	11498	83855
TF-IDF	-	0.422	0.300	0.153	0.182

Table 3: Statistics of SemCor per POS

statistics of SemCor is listed in table 3.

We use hyperparameters tuned from the text categorization task: α_d=0.1, α_s=0.01, β=0.01, δ=1, T=50, and try different values of $\epsilon \in \{10, 20, 40\}$ and $\gamma \in \{0, 2, 10\}$. The Brown corpus and WordNet definitions corpus are used as augmented data, which means the dashed line in figure 1c will become bold. Finally, we choose the most frequent answer for each word in the last 10 iterations of a Gibbs Sampling run as the final sense choice.

WSD Results: Disambiguation per POS results are presented in table 4. We only report results on polysemous words. We can see that modeling definitions (STM2 and STM10) improves performance significantly over STM0's across the board per POS and overall. The fact that STMn picks more correct senses helps explain why STMn classifies more documents correctly than STM0. Also it is interesting to see that unlike in the text categorization task, larger values of γ generate better WSD results. However, the window size ϵ, does not make a significant difference, yet we note that ϵ=10 is still the optimal value, similar to our observation in the text categorization task.

STM10 achieves similar results as in LDAWN (Boyd-Graber et al., 2007) which was specifically designed for WSD. LDAWN needs a fine grained hypernym hierarchy to perform WSD, hence they can only disambiguate nouns. They report different performances under various parameter setting. We cite their best performance of 38% accuracy on nouns as a comparison point to our best performance for nouns of 38.5%.

An interesting feature of STM10 is that it performs much better in nouns than adverbs and verbs, compared to a random baseline in Table 4. This is understandable since topic information content is mostly borne by nouns and adjectives, while adverbs and verbs tend to be less informative about topics (e.g., *even, indicate, take*), and used more across different domain documents. Hence topic models are weaker in their ability to identify clear cues for senses for verbs and adverbs. In support of our hypothesis about the POS distribution, we compute the average TF-IDF scores for each POS (shown in Table 3 according to the equation illustrated below). The average TF-IDF clearly indicate the positive skewness of the nouns and adjectives (high TF-IDF) correlates with the better WSD performance.

$$\text{TF-IDF}(pos) = \frac{\sum_i \sum_d \text{TF-IDF}(w_{i,d})}{\#\ of\ w_{i,d}}$$
$$where\ w_{i,d} \in pos.$$

At last, we notice that the most frequent sense baseline performs much better than our models. This is understandable since: (1) most frequent sense baseline can be treated as a supervised method in the sense that the sense frequency is calculated based on the sense choice as present in sense annotated data; (2) our model is not designed for WSD, therefore it discards a lot of information when choosing the sense: in our model, the choice of a sense s_i is only dependent on two facts: the corresponding topic z_i and word w_i, while in (Li et al., 2010; Banerjee and Pedersen, 2003), they consider all the senses and words in the context words.

5 Related work

Various topic models have been developed for many applications. Recently there is a trend of modeling document dependency (Dietz et al., 2007; Mei et al., 2008; Daume, 2009). However, topics are only inferred based on word cooccurrence, while word semantics are ignored.

Boyd-Graber et al. (2007) are the first to integrate semantics into the topic model framework. They propose a topic model based on WordNet noun hierarchy for WSD. A word is assumed to be generated by first sampling a topic, then choosing a path from the root node of hierarchy to a sense node corresponding to that word. However, they only focus on WSD. They do not exploit word definitions, neither do they report results on text categorization.

Chemudugunta et al. (2008) also incorporate a sense hierarchy into a topic model. In their framework, a word may be directly generated from a topic (as in standard topic models), or it can be

		Total	Noun	Adjective	Adverb	Verb
random		22.1	26.2	27.9	32.2	15.8
most frequent sense		64.7	74.7	77.5	74.0	59.6
STM0	$\epsilon = 10$	24.1±1.4	29.3±4.3	28.7±1.1	34.1±3.1	17.1±1.6
	$\epsilon = 20$	24±1.3	30.2±3.3	29.1±1.4	34.9±3.1	15.9±0.7
	$\epsilon = 40$	24±2.4	28.4±4.3	28.7±1.1	**36.4±4.7**	17.3±2.4
STM2	$\epsilon = 10$	27.5±1.1	36.1±3.8	34.0±1.2	33.4±1.8	17.8±1.4
	$\epsilon = 20$	25.7±1.3	32.0±4.2	33.5±0.7	34.2±3.4	17.3±0.7
	$\epsilon = 40$	26.1±1.3	32.5±3.9	33.6±0.9	34.2±3.4	17.5±1.4
STM10	$\epsilon = 10$	**28.8±1.1**	**38.5±2.3**	**34.7±0.8**	34.0±3.3	**18.4±1.2**
	$\epsilon = 20$	27.7±1.0	36.8±2.2	34.5±0.7	33.0±3.1	17.6±0.7
	$\epsilon = 40$	28.1±1.5	38.4±3.1	34.0±1.0	35.1±5.4	17.0±0.9

Table 4: Disambiguation results per POS on polysemous words.

generated by choosing a sense path in the hierarchy. Note that no topic information is on the sense path. If a word is generated from the hierarchy, then it is not assigned a topic. Their models based on different dictionaries improve perplexity.

Recently, several systems have been proposed to apply topic models to WSD. Cai et al. (2007) incorporate topic features into a supervised WSD framework. Brody and Lapata (2009) place the sense induction in a Baysian framework by assuming each context word is generated from the target word's senses, and a context is modeled as a multinomial distribution over the target word's senses rather than topics. Li et al. (2010) design several systems that use latent topics to find a most likely sense based on the sense paraphrases (extracted from WordNet) and context. Their WSD models are unsupervised and outperform state-of-art systems.

Our model borrows the local window idea from word sense disambiguation community. In graph-based WSD systems (Mihalcea, 2005; Sinha and Mihalcea, 2007; Guo and Diab, 2010), a node is created for each sense. Two nodes will be connected if their distance is less than a predefined value; the weight on the edge is a value returned by sense similarity measures, then the PageRank/Indegree algorithm is applied on this graph to determine the appropriate senses.

6 Conclusion and Future Work

We presented a novel model STM that combines explicit semantic information and word distribution information in a unified topic model. STM is able to capture topics of words more accurately than traditional LDA topic models. In future work, we plan to model the WordNet sense network. We believe that WordNet senses are too fine-grained, hence we plan to use clustered senses, instead of current WN senses, in order to avail the model of more generalization power.

Acknowledgments

This research was funded by the Ofce of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the U.S. Army Research Lab. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the ofcial views or policies of IARPA, the ODNI or the U.S. Government.

References

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 805–810.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber and David M. Blei. 2007. Putop: turning predominant senses into a topic model for word sense disambiguation. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 277–281.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1024–1033.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 103–111.

Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. 2007. Improving word sense disambiguation using topic features. In *Proceedings of 2007 Joint Conference on Empirical Methods in Natural Language*

Processing and Computational Natural Language Learning, pages 1015–1023.

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2008. Combining concept hierarchies and statistical topic models. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1469–1470.

Hal Daume. 2009. Markov random topic fields. In *Proceedings of the ACL-IJCNLP Conference*, pages 293–296.

Laura Dietz, Steffen Bickel, and Tobias Scheffer. 2007. Unsupervised prediction of citation influence. In *Proceedings of the 24th international conference on Machine learning*, pages 233–240.

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

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *Advances in Neural Information Processing Systems*.

Weiwei Guo and Mona Diab. 2010. Combining orthogonal monolingual and multilingual sources of evidence for all words wsd. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1542–1551.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1138–1147.

Qiaozhu Mei, Deng Cai, Duo Zhang, and Chengxiang Zhai. 2008. Topic modeling with network regularization. In *Proceedings of the 17th international conference on World Wide Web*, pages 101–110.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 411–418.

Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 task 17: English lexical sample, srl and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 87–92. ACL.

Ravi Sinha and Rada Mihalcea. 2007. Unsupervised graph-based word sense disambiguation using measures of word semantic similarity. In *Proceedings of the IEEE International Conference on Semantic Computing*, pages 363–369.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43. ACL.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120.

Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs

Samuel Brody
School of Communication
and Information
Rutgers University
sdbrody@gmail.com

Nicholas Diakopoulos
School of Communication
and Information
Rutgers University
diakop@rutgers.edu

Abstract

We present an automatic method which leverages word lengthening to adapt a sentiment lexicon specifically for Twitter and similar social messaging networks. The contributions of the paper are as follows. First, we call attention to lengthening as a widespread phenomenon in microblogs and social messaging, and demonstrate the importance of handling it correctly. We then show that lengthening is strongly associated with subjectivity and sentiment. Finally, we present an automatic method which leverages this association to detect domain-specific sentiment- and emotion-bearing words. We evaluate our method by comparison to human judgments, and analyze its strengths and weaknesses. Our results are of interest to anyone analyzing sentiment in microblogs and social networks, whether for research or commercial purposes.

1 Introduction

Recently, there has been a surge of interest in sentiment analysis of Twitter messages. Many researchers (e.g., Bollen et al. 2011; Kivran-Swaine and Naaman 2011) are interested in studying structure and interactions in social networks, where sentiment can play an important role. Others use Twitter as a barometer for public mood and opinion in diverse areas such as entertainment, politics and economics. For example, Diakopoulos and Shamma (2010) use Twitter messages posted in conjunction with the live presidential debate between Barack Obama and John McCain to gauge public opinion, Bollen et al. (2010) measure public mood on Twitter and use it to predict upcoming stock market fluctuations, and O'Connor et al. (2010) connect public opinion data from polls to sentiment expressed in Twitter messages along a timeline.

A prerequisite of all such research is an effective method for measuring the sentiment of a post or *tweet*. Due to the extremely informal nature of the medium, and the length restriction[1], the language and jargon which is used in Twitter varies significantly from that of commonly studied text corpora. In addition, Twitter is a quickly evolving domain, and new terms are constantly being introduced. These factors pose difficulties to methods designed for conventional domains, such as news. One solution is to use human annotation. For example, Kivran-Swaine and Naaman (2011) use manual coding of tweets in several emotion categories (e.g., joy, sadness) for their research. Diakopoulos and Shamma (2010) use crowd sourcing via Amazon's Mechanical Turk. Manual encoding usually offers a deeper understanding and correspondingly higher accuracy than shallow automatic methods. However, it is expensive and labor intensive and cannot be applied in real time. Crowd-sourcing carries additional caveats of its own, such as issues of annotator expertise and reliability (see Diakopoulos and Shamma 2010).

The automatic approach to sentiment analysis is commonly used for processing data from social networks and microblogs, where there is often a huge quantity of information and a need for low latency. Many automatic approaches (including all those used in the work mentioned above) have at their core a sentiment lexicon, containing a list of words la-

[1] Messages in Twitter are limited to 140 characters, for compatibility with SMS messaging

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 562–570,
Edinburgh, Scotland, UK, July 27–31, 2011.

beled with specific associated emotions (joy, happiness) or a polarity value (positive, neutral, negative). The overall sentiment of a piece of text is calculated as a function of the labels of the component words. Because Twitter messages are short, shallow approaches are sometimes considered sufficient (Bermingham and Smeaton, 2010). There are also approaches that use deeper machine learning techniques to train sentiment classifiers on examples that have been labeled for sentiment, either manually or automatically, as described above. Recent examples of this approach are Barbosa and Feng (2010) and Pak and Paroubek (2010).

Most established sentiment lexicons (e.g., Wilson et al. 2005, see Section 5) were created for a general domain, and suffer from limited coverage and inaccuracies when applied to the highly informal domain of social networks communication. By creating a sentiment lexicon which is specifically tailored to the microblogging domain, or adapting an existing one, we can expect to achieve higher accuracy and increased coverage. Recent work in this area includes Velikovich et al. (2010), who developed a method for automatically deriving an extensive sentiment lexicon from the web as a whole. The resulting lexicon has greatly increased coverage compared to existing dictionaries and can handle spelling errors and web-specific jargon. Bollen et al. (2010) expand an existing well-vetted psychometric instrument - Profile of Mood States (POMS) (McNair et al., 1971) that associates terms with moods (e.g. calm, happy). The authors use co-occurrence information from the Google n-gram corpus (Brants and Franz, 2006) to enlarge the original list of 72 terms to 964. They use this expanded emotion lexicon (named GPOS) in conjunction with the lexicon of Wilson et al. (2005) to estimate public mood from Twitter posts[2].

The method we present in this paper leverages a phenomenon that is specific to informal social communication to enable the extension of an existing lexicon in a domain specific manner.

[2]Although the authors state that all data and methods will be made available on a public website, it was not present at the time of the writing of this article.

2 Methodology

Prosodic indicators (such as high pitch, prolonged duration, intensity, vowel quality, etc.) have long been know (Bolinger, 1965) as ways for a speaker to emphasize or accent an important word. The ways in which they are used in speech are the subject of ongoing linguistic research (see, for example, Calhoun 2010). In written text, many of these indicators are lost. However, there exist some orthographic conventions which are used to mark or substitute for prosody, including punctuation and typographic styling (italic, bold, and underlined text). In purely text-based domains, such as Twitter, styling is not always available, and is replaced by capitalization or other conventions (e.g., enclosing the word in asterisks). Additionally, the informal nature of the domain leads to an orthographic style which is much closer to the spoken form than in other, more formal, domains. In this work, we hypothesize that the commonly observed phenomenon of lengthening words by repeating letters is a substitute for prosodic emphasis (increased duration or change of pitch). As such, it can be used as an indicator of important words and, in particular, ones that bear strong indication of sentiment.

Our experiments are designed to analyze the phenomenon of lengthening and its implications to sentiment detection. First, in Experiment I, we show the pervasiveness of the phenomenon in our dataset, and measure the potential gains in coverage that can be achieved by considering lengthening when processing Twitter data. Experiment II substantiates the claim that word lengthening is not arbitrary, and is used for emphasis of important words, including those conveying sentiment and emotion. In the first part of Experiment III we demonstrate the implications of this connection for the purpose of sentiment detection using an existing sentiment lexicon. In the second part, we present an unsupervised method for using the lengthening phenomenon to expand an existing sentiment lexicon and tailor it to our domain. We evaluate the method through comparison to human judgments, analyze our results, and demonstrate some of the benefits of our automatic method.

1. For every word in the vocabulary, extract the condensed form, where sequences of a repeated letter are replaced with a single instance of that letter.
   E.g., $niiiice \rightarrow nice,\ reallllly \rightarrow realy$ ...
2. Create sets of words sharing the same condensed form.
   E.g., $\{nice, niiice, niccccceee...\},\ \{realy, really, reallllly, ...\}$ ...
3. Remove sets which do not contain at least one repeat of length three.
   E.g.,$\{committee, committe, commitee\}$
4. Find the most frequently occurring form in the group, and mark it as the canonical form.
   E.g., $\{$**nice**$, niiice, niccccceee...\},\ \{realy,$ **really**$, reallllly, ...\}$ ...

Figure 1: Procedure for detecting lengthened words and associating them with a canonical form.

3 Data

Half a million tweets were sampled from the Twitter Streaming API on March 9th 2011. The tweets were sampled to cover a diverse geographic distribution within the U.S. such that regional variation in language use should not bias the data. Some tweets were also sampled from Britain to provide a more diverse sampling of English. We restricted our sample to tweets from accounts which indicated their primary language as English. However, there may be some foreign language messages in our dataset, since multi-lingual users may tweet in other languages even though their account is marked as "English".

The tweets were tokenized and converted to lower-case. Punctuation, as well as links, hashtags, and username mentions were removed. The resulting corpus consists of approximately 6.5 million words, with a vocabulary of 22 thousand words occurring 10 times or more.

4 Experiment I - Detection

To detect and analyze lengthened words, we employ the procedure described in Figure 1. We find sets of words in our data which share a common form and differ only in the number of times each letter is repeated (Steps 1 & 2). In Step 3 we remove sets where all the different forms are likely to be the result of misspelling, rather than lengthening. Finally, in Step 4, we associate all the forms in a single set with a canonical form, which is the most common one observed in the data.

The procedure resulted in 4,359 sets of size > 1. To reduce noise resulting from typos and misspellings, we do not consider words containing non-alphabetic characters, or sets where the canonical form is a single character or occurs less than 10 times. This left us with 3,727 sets.

Analysis Table 1 lists the canonical forms of the 20 largest sets in our list (in terms of the number of variations). Most of the examples are used to express emotion or emphasis. Onomatopoeic words expressing emotion (e.g., *ow, ugh, yay*) are often lengthened and, for some, the combined frequency of the different lengthened forms is actually greater than that of the canonical (single most frequent) one.

Lengthening is a common phenomenon in our dataset. Out of half-a-million tweets, containing roughly 6.5 million words, our procedure identifies 108,762 word occurrences which are lengthenings of a canonical form. These words occur in 87,187 tweets (17.44% or approximately one out of every six, on average). The wide-spread use of lengthening is surprising in light of the length restriction of Twitter. Grinter and Eldridge (2003) point out several conventions that are used in text messages specifically to deal with this restriction. The fact that lengthening is used in spite of the need for brevity suggests that it conveys important information.

Canonical Assumption We validate the assumption that the most frequent form in the set is the canonical form by examining sets containing one or more word forms that were identified in a standard

Can. Form	Card.	# Can.	# Non-Can.
nice	76	3847	348
ugh	75	1912	1057
lmao	70	10085	3727
lmfao	67	2615	1619
ah	61	767	1603
love	59	16360	359
crazy	59	3530	253
yeah	57	4562	373
sheesh	56	247	131
damn	52	5706	299
shit	51	10332	372
really	51	9142	142
oh	51	7114	1617
yay	45	1370	375
wow	45	3767	223
good	45	21042	3171
ow	44	116	499
mad	44	3627	827
hey	44	4669	445
please	43	4014	157

Table 1: The canonical forms of the 20 largest sets (in terms of cardinality), with the number of occurrences of the canonical and non-canonical forms.

English dictionary[3]. This was the case for 2,092 of the sets (56.13%). Of these, in only 55 (2.63%) the most frequent form was *not* recognized by the dictionary. This indicates that the strategy of choosing the most frequent form as the canonical one is reliable and highly accurate ($> 97\%$).

Implications for NLP To examine the effects of lengthening on analyzing Twitter data, we look at the difference in coverage of a standard English dictionary when we explicitly handle lengthened words by mapping them to the canonical form. Coverage with a standard dictionary is important for many NLP applications, such as information retrieval, translation, part-of-speech tagging and parsing. The canonical form for 2,037 word-sets are identified by our dictionary. We searched for occurrences of these words which were lengthened by two or more characters, meaning they would not be identified using standard lemmatization methods or spell-correction techniques that are based on edit distance. We detected 25,101 occurrences of these, appearing in 22,064 (4.4%) tweets. This implies that a lengthening-aware stemming method can be used to increase coverage substantially.

[3]We use the standard dictionary for U.S. English included in the Aspell Unix utility.

5 Experiment II - Relation to Sentiment

At the beginning of Section 2 we presented the hypothesis that lengthening represents a textual substitute for prosodic indicators in speech. As such, it is not used arbitrarily, but rather applied to subjective words to strengthen the sentiment or emotion they convey. The examples presented in Table 1 in the previous section appear to support this hypothesis. In this section we wish to provide experimental evidence for our hypothesis, by demonstrating a significant degree of association between lengthening and subjectivity.

For this purpose we use an existing sentiment lexicon (Wilson et al., 2005), which is commonly used in the literature (see Section 1) and is at the core of OpinionFinder[4], a popular sentiment analysis tool designed to determine opinion in a general domain. The lexicon provides a list of subjective words, each annotated with its degree of subjectivity (strongly subjective, weakly subjective), as well as its sentiment polarity (positive, negative, or neutral). In these experiments, we use the presence of a word (canonical form) in the lexicon as an indicator of subjectivity. It should be noted that the reverse is not true, i.e., the fact that a word is absent from the lexicon does not indicate it is objective.

As a measure of tendency to lengthen a word, we look at the number of distinct forms of that word appearing in our dataset (the cardinality of the set to which it belongs). We group the words according to this statistic, and compare to the vocabulary of our dataset (all words appearing in our data ten times or more, and consisting of two or more alphabetic characters, see Section 4). Figure 2 shows the percentage of subjective words (those in the lexicon) in each of the groups. As noted previously, this is a lower bound, since it is possible (in fact, very likely) that other words in the group are subjective, despite being absent from the lexicon. The graph shows a clear trend - the more lengthening forms a word has,

[4]http://www.cs.pitt.edu/mpqa/opinionfinderrelease/

the more likely it is to be subjective (as measured by the percentage of words in the lexicon).

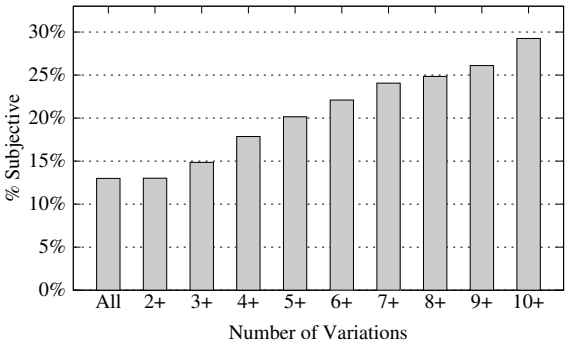
The reverse also holds - if a word is used to convey sentiment, it is more likely to be lengthened. We can verify this by calculating the average number of distinct forms for words in our data that are subjective and comparing to the rest. This calculation yields an average of 2.41 forms for words appearing in our sentiment lexicon (our proxy for subjectivity), compared to an average of 1.79 for those that aren't[5]. This difference is statistically significant at $p < 0.01\%$, using a student t-test.

The lexicon we use was designed for a general domain, and suffers from limited coverage (see below) and inaccuracies (see O'Connor et al. 2010 and below Section 6.2 for examples), due to the domain shift. The sentiment lexicon contains 6,878 words, but only 4,939 occur in our data, and only 2,446 appear more than 10 times. Of those appearing in our data, only 485 words (7% of the lexicon vocabulary) are lengthened (the bar for group 2+ in Figure 2), but these are extremely salient. They encompass 701,607 instances (79% of total instances of words from the lexicon), and 339,895 tweets. This provides further evidence that lengthening is used with salient sentiment words.

These results also demonstrates the limitations of using a sentiment lexicon which is not tailored to the domain. Only a small fraction of the lexicon is represented in our data, and it is likely that there are many sentiment words that are commonly used but are absent from it. We address this issue in the next section.

6 Experiment III - Adapting the Sentiment Lexicon

The previous experiment showed the connection between lengthening and sentiment-bearing words. It also demonstrated some of the shortcomings of a lexicon which is not specifically tailored to our domain. There are two steps we can take to use the lengthening phenomenon to adapt an existing sentiment lexicon. The first of these is simply to take lengthening into account when identifying sentiment-bearing words in our corpus. The second is to exploit the connection between lengthening and sentiment to expand the lexicon itself.



All	2+	3+	4+	5+	6+	7+	8+	9+	10+
18,817	3,727	2,451	1,540	1,077	778	615	487	406	335

Figure 2: The percentage of subjective word-sets (those whose canonical form appears in the lexicon) as a function of cardinality (number of lengthening variations). The accompanying table provides the total number of sets in each cardinality group.

6.1 Expanding Coverage of Existing Words

We can assess the effect of specifically considering lengthening in our domain by measuring the increase of coverage of the existing sentiment lexicon. Similarly to Experiment I (Section 4), we searched for occurrences of words from the lexicon which were lengthened by two or more characters, and would therefore not be detected using edit-distance. We found 12,105 instances, occurring in 11,439 tweets (2.29% of the total). This increase in coverage is relatively small[6], but comes at almost no cost, by simply considering lengthening in the analysis.

A much greater benefit of lengthening, however, results from using it as an aid in expanding the sentiment lexicon and detecting new sentiment-bearing words. This is the subject of the following section.

6.2 Expanding the Sentiment Vocabulary

In Experiment II (Section 5) we showed that lengthening is strongly associated with sentiment. Therefore, words which are lengthened can provide us with good candidates for inclusion in the lexicon. We can employ existing sentiment-detection meth-

[5]This, too, is a conservative estimate, since the later group also includes subjective words, as mentioned.

[6]Note that almost half of the increase in coverage calculated in Experiment I (Section 4) comes from subjective words!

ods to decide which candidates to include, and determine their polarity.

Choosing a Candidate Set The first step in expanding the lexicon is to choose a set of candidate words for inclusion. For this purpose we start with words that have 5 or more distinct forms. There are 1,077 of these, of which only 217 (20.15%) are currently in our lexicon (see Figure 2). Since we are looking for commonly lengthened words, we disregard those where the combined frequency of the non-canonical forms is less than 1% that of the canonical one. We also remove stop words, even though some are often lengthened for emphasis (e.g., *me, and, so*), since they are too frequent, and introduce many spurious edges in our co-occurrence graph. Finally, we filter words based on weight, as described below. This leaves us with 720 candidate words.

Graph Approach We examine two methods for sentiment detection - that of Brody and Elhadad (2010) for detecting sentiment in reviews, and that of Velikovich et al. (2010) for finding sentiment terms in a giga-scale web corpus. Both of these employ a graph-based approach, where candidate terms are nodes, and sentiments is propagated from a set of seed words of known sentiment polarity. We calculated the frequency in our corpus of all strongly positive and strongly negative words in the Wilson et al. (2005) lexicon, and chose the 100 most frequent in each category as our seed sets.

Graph Construction Brody and Elhadad (2010) considered all frequent adjectives as candidates and weighted the edge between two adjectives by a function of the number of times they both modified a single noun. Velikovich et al. (2010) constructed a graph where the nodes were 20 million candidate words or phrases, selected using a set of heuristics including frequency and mutual information of word boundaries. Context vectors were constructed for each candidate from all its mentions in a corpus of 4 billion documents, and the edge between two candidates was weighted by the cosine similarity between their context vectors.

Due to the nature of the domain, which is highly informal and unstructured, accurate parsing is difficult. Therefore we cannot employ the exact construction method of Brody and Elhadad (2010). On the other hand, the method of Velikovich et al. (2010) is based on huge amounts of data, and takes advantage of the abundance of contextual information available in full documents, whereas our domain is closer to that of Brody and Elhadad (2010), who dealt with a small number of candidates and short documents typical to online reviews. Therefore, we adapt their construction method. We consider all our candidates words as nodes, along with the words in our positive and negative seed sets. As a proxy for syntactic relationship, edges are weighted as a function of the number of times two words occurred within a three-word window of each other in our dataset. We remove nodes whose neighboring edges have a combined weight of less than 20, meaning they participate in relatively few co-occurrence relations with the other words in the graph.

Algorithm Once the graph is constructed, we can use either of the propagation algorithms of Brody and Elhadad (2010) and Velikovich et al. (2010), which we will denote Reviews and Web, respectively. The Reviews propagation method is based on Zhu and Ghahramani (2002). The words in the positive and negative seed groups are assigned a polarity score of 1 and 0, respectively. All the rest start with a score of 0.5. Then, an update step is repeated. In update iteration t, for each word x that is *not in the seed*, the following update rule is applied:

$$p^t(x) = \frac{\Sigma_{y \in N(x)} w(y,x) \cdot p^{t-1}(y)}{\Sigma_{y \in N(x)} w(y,x)} \qquad (1)$$

Where $p^t(x)$ is the polarity of word x at step t, $N(x)$ is the set of the neighbors of x, and $w(y,x)$ is the weight of the edge connecting x and y. Following Brody and Elhadad (2010), we set this weight to be $1 + log(\#co(y,x))$, where $\#co(y,x)$ is the number of times y and x co-occurred within a three-word window. The update step is repeated to convergence.

Velikovich et al. (2010) employed a different label propagation method, as described in Figure 3. Rather than relying on diffusion along the whole graph, this method considers only the single strongest path between each candidate and each seed word. In their paper, the authors claim that their algorithm is more suitable than that of Zhu and Ghahramani (2002) to a web-based dataset, which

Input: $G = (V, E), w_{ij} \in [0, 1]$
$P, N, \gamma \in \mathbb{R}, T \in \mathbb{N}$
Output: $\text{pol}_i \in \mathbb{R}^{|V|}$
Initialize: $\text{pol}_i, \text{pol}_i^+, \text{pol}_i^- = 0$ for all i
$\text{pol}_i^+ = 1.0$ for all $v_i \in P$ and
$\text{pol}_i^- = 1.0$ for all $v_i \in N$

1: $\alpha_{ij} = 0$ for all $i \neq j$, $\alpha_{ii} = 1$ for all i
2: **for** $v_i \in P$
3: $F = \{v_i\}$
4: **for** $t : 1...T$
5: **for** $(v_k, v_j) \in E$ such that $v_k \in F$
6: $\alpha_{ij} = max(\alpha_{ij}, \alpha_{ik} \cdot w_{k,j})$
$F = F \cup \{v_j\}$
7: **for** $v_j \in V$
8: $\text{pol}_j^+ = \sum_{v_i \in P} \alpha_{ij}$
9: Repeat steps 1-8 using N to compute pol^-
10: $\beta = \sum_i \text{pol}_i^+ / \sum_i \text{pol}_i^-$
11: $\text{pol}_i = \text{pol}_i^+ - \beta \text{pol}_i^-$, for all i
12: if $|\text{pol}_i| < \gamma$ then $\text{pol}_i = 0.0$ for all i

Figure 3: Web algorithm from Velikovich et al. (2010). P and N are the positive and negative seed sets, respectively, w_{ij} are the weights, and T and γ are parameters[9].

contained many dense subgraphs and unreliable associations based only on co-occurrence statistics. We ran both algorithms in our experiment[7], and compared the results.

Evaluation We evaluated the output of the algorithms by comparison to human judgments. For words appearing in the sentiment lexicon, we used the polarity label provided. For the rest, similarly to Brody and Elhadad (2010), we asked volunteers to rate the words on a five-point scale: *strongly-negative, weakly-negative, neutral, weakly-positive,* or *strongly-positive*. We also provided a *N/A* option if the meaning of the word was unknown. Each word was rated by two volunteers. Words which were labeled *N/A* by one or more annotators were considered *unknown*. For the rest, exact inter-rater agreement was 67.6%, but rose to 93% when considering adjacent ratings as equivalent[10]. This is comparable with the agreement reported by Brody and Elhadad (2010). We assigned values 1 (strongly negative) to 5 (strongly positive) to the ratings, and calculated the average between the two ratings for each word. Words with an average rating of 3 were considered neutral, and those with lower and higher ratings were considered negative and positive, respectively.

[7]We normalize the weights described above when using the Web algorithm.

[9]In Velikovich et al. (2010), the parameters T and γ were tuned on a held out dataset. Since our graphs are comparatively small, we do not need to limit the path length T in our search. We do not use the threshold γ, but rather employ a simple cutoff of the top 50 words.

		Human Judgment			
		Pos.	Neg.	Neu.	Unk.
Web	Pos.	18	2	26	2
	Neg.	8	19	17	8
Reviews	Pos.	21	6	21	2
	Neg.	9	14	11	16

Table 2: Evaluation of the top 50 positive and negative words retrieved by the two algorithms through comparison to human judgment.

Web		Reviews	
pos.	neg.	pos.	neg.
see	shit	kidding	rell
win	niggas	justin	whore
way	dis	win	rocks
gotta	gettin	feel	ugg
summer	smh	finale	naw
lets	tight	totally	yea
haha	fuckin	awh	headache
birthday	fuck	boys	whack
tomorrow	sick	pls	yuck
ever	holy	ever	shawty
school	smfh	yer	yeah
peace	outta	lord	sus
soon	odee	mike	sleepy
stuff	wack	three	hunni
canes	nigga	agreed	sick

Table 3: Top fifteen negative and positive words for the algorithms of Brody and Elhadad (2010) (Reviews) and Velikovich et al. (2010) (Web).

Results Table 2 shows the distribution of the human labels among the top 50 most positive and most negative words as determined by the two algorithms. Table 3 lists the top 15 of these as examples.

[10]Cohen's Kappa $\kappa = 0.853$

From Table 2 we can see that both algorithms do better on positive words (fewer words with reversed polarity)[11], and that the Web algorithm is more accurate than the Reviews method. The difference in performance can be explained by the associations used by the algorithms. The Web algorithm takes into account the strongest path to *every* seed word, while the Reviews algorithm propagates from the each seed to its neighbors and then onward. This makes the Reviews algorithm sensitive to strong associations between a word and a single seed. Because our graph is constructed with co-occurrence edges between words, rather than syntactic relations between adjectives, noisy edges are introduced, causing mistaken associations. The Web algorithm, on the other hand, finds words that have a strong association with the positive or negative seed group as a whole, thus making it more robust. This explains some of the examples in Table 3. The words *yeah* and *yea*, which often follow the negative seed word *hell*, are considered negative by the Reviews algorithm. The word *Justin* refers to Justin Beiber, and is closely associated with the positive seed word *love*. Although the Web algorithm is more robust to associations with a single seed, it still misclassifies the word *holy* as negative, presumably because it appears frequently before several different expletives.

Detailed analysis shows that the numbers reported in Table 2 are only rough estimates of performance. For instance, several of the words in the *unknown* category were correctly identified by the algorithm. Examples include *sm(f)h*, which stands for *"shaking my (fucking) head"* and expresses disgust or disdain, *sus*, which is short for *suspicious* (as in *"i hate susssss ass cars that follow me/us when i'm/we walkinggg"*), and *odee*, which means *overdose* and is usually negative (though it does not always refers to drugs, and is sometimes used as an intensifier, e.g., *"aint shit on tv odee bored"*).

There were also cases were the human labels were incorrect in the context of our domain. For example, the word *bull* is listed as positive in the sentiment lexicon, presumably because of its financial sense. In our domain it is (usually) short for *bullshit*. The word *canes* was rated as negative by one of the annotators, but in our data it refers to the Miami Hurricanes, who won a game on the day our dataset was sampled, and were the subject of many positive tweets. This example also demonstrates that our method is capable of detecting terms which are associated with sentiment at different time points, something that is not possible with a fixed lexicon.

[11] This trend is not apparent from the top 15 results presented in Table 3, but becomes noticeable when considering the larger group.

7 Conclusion

In this paper we explored the phenomenon of lengthening words by repeating a single letter. We showed that this is a common phenomenon in Twitter, occurring in one of every six tweets, on average, in our dataset. Correctly detecting these cases is important for comprehensive coverage. We also demonstrated that lengthening is not arbitrary, and is often used with subjective words, presumably to emphasize the sentiment they convey. This finding leads us to develop an unsupervised method based on lengthening for detecting new sentiment bearing words that are not in the existing lexicon, and discovering their polarity. In the rapidly-changing domain of microblogging and net-speak, such a method is essential for up-to-date sentiment detection.

8 Future Work

This paper examined one aspect of the lengthening phenomenon. There are other aspects of lengthening that merit research, such as the connection between the amount of lengthening and the strength of emphasis in individual instances of a word. In addition to sentiment-bearing words, we saw other word classes that were commonly lengthened, including intensifiers (e.g., *very, so, odee*), and named entities associated with sentiment (e.g., *Justin, 'Canes*). These present interesting targets for further study. Also, in this work we focused on data in English, and it would be interesting to examine the phenomenon in other languages. Another direction of research is the connection between lengthening and other orthographic conventions associated with sentiment and emphasis, such as emoticons, punctuation, and capitalization. Finally, we plan to integrate lengthening and its related phenomena into an accurate, Twitter-specific, sentiment classifier.

Acknowledgements

The authors would like to thank Paul Kantor and Mor Naaman for their support and assistance in this project. We would also like to thank Mark Steedman for his help, and the anonymous reviewers for their comments and suggestions.

References

Barbosa, Luciano and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING (Posters)*. Chinese Information Processing Society of China, pages 36–44.

Bermingham, Adam and Alan F. Smeaton. 2010. Classifying sentiment in microblogs: is brevity an advantage? In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, New York, NY, USA, CIKM '10, pages 1833–1836.

Bolinger, Dwight. 1965. *Forms of English: Accent, Morpheme, Order*. Harvard University Press, Cambridge, Massachusetts, USA.

Bollen, J., H. Mao, and X.-J. Zeng. 2010. Twitter mood predicts the stock market. *ArXiv e-prints* .

Bollen, Johan, Bruno Gonalves, Guangchen Ruan, and Huina Mao. 2011. Happiness is assortative in online social networks. *Artificial Life* 0(0):1–15.

Brants, Thorsten and Alex Franz. 2006. Google web 1T 5-gram corpus, version 1. Linguistic Data Consortium, Catalog Number LDC2006T13.

Brody, Samuel and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2010)*. ACL, Los Angeles, CA, pages 804–812.

Calhoun, Sasha. 2010. The centrality of metrical structure in signaling information structure: A probabilistic perspective. *Language* 86:1–42.

Diakopoulos, Nicholas A. and David A. Shamma. 2010. Characterizing debate performance via aggregated twitter sentiment. In *Proceedings of the 28th international conference on Human factors in computing systems*. ACM, New York, NY, USA, CHI '10, pages 1195–1198.

Grinter, Rebecca and Margery Eldridge. 2003. Wan2tlk?: everyday text messaging. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, New York, NY, USA, CHI '03, pages 441–448.

Kivran-Swaine, Funda and Mor Naaman. 2011. Network properties and social sharing of emotions in social awareness streams. In *Proceedings of the 2011 ACM Conference on Computer Supported Cooperative Work (CSCW 2011)*. Hangzhou, China.

McNair, D. M., M. Lorr, and L. F. Droppleman. 1971. *Profile of Mood States (POMS)*. Educational and Industrial Testing Service.

O'Connor, Brendan, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.

Pak, Alexander and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. ELRA, Valletta, Malta.

Velikovich, Leonid, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. ACL, Stroudsburg, PA, USA, HLT '10, pages 777–785.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. ACL, Stroudsburg, PA, USA, HLT '05, pages 347–354.

Zhu, X. and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02.

Personalized Recommendation of User Comments via Factor Models

Deepak Agarwal **Bee-Chung Chen** **Bo Pang**
Yahoo! Research
701 First Ave
Sunnyvale, CA 94089
{dagarwal,beechun,bopang}@yahoo-inc.com

Abstract

In recent years, the amount of user-generated opinionated texts (e.g., reviews, user comments) continues to grow at a rapid speed: featured news stories on a major event easily attract thousands of user comments on a popular online News service. How to consume subjective information of this volume becomes an interesting and important research question. In contrast to previous work on review analysis that tried to filter or summarize information for a generic *average* user, we explore a different direction of enabling personalized recommendation of such information.

For each user, our task is to rank the comments associated with a given article according to personalized user preference (i.e., whether the user is likely to like or dislike the comment). To this end, we propose a factor model that incorporates rater-comment and rater-author interactions simultaneously in a principled way. Our full model significantly outperforms strong baselines as well as related models that have been considered in previous work.

1 Introduction

Recent years have seen rapid growth in user-generated opinions online. Many of them are user reviews: a best-seller or a popular restaurant can get over 1000 reviews on top review sites like Amazon or Yelp. A large quantity of them also come in the form of user comments on blogs or news articles. Most notably, during the short period of time for which a major event is active, news stories on one single event can easily attract over ten thousand comments on a popular online news site like Yahoo! News. One question becomes immediate: how can we help people consume such gigantic amount of opinionated information?

One possibility is to take the summarization route. Briefly speaking (see Section 2 for a more detailed discussion), previous work has largely formulated review summarization as automatically or manually identify ratable aspects, and present overall sentiment polarity for each aspect (Hu and Liu, 2004; Popescu and Etzioni, 2005; Snyder and Barzilay, 2007; Titov and McDonald, 2008). A related line of research looked into predicting helpfulness of reviews in the hope of promoting those with better quality, where helpfulness is usually defined as some function over the percentage of users who found the review to be helpful (Kim et al., 2006; Liu et al., 2007; Danescu-Niculescu-Mizil et al., 2009). In short, the focus of previous work has been on distilling subjective information for an *average* user.

Whether opinion consumers are looking for quality information or just wondering what other people think, each may have different purposes or preferences that is not well represented by a generic average user. If we think about how we deal with information content overflow on the Web, there have been two main frameworks to identify relevant information for each person. One is search. Indeed many top review sites allow users to search within reviews for a given entity. But this is only useful when users have explicit information needs that can be formulated as queries. The other paradigm is recommendation: based on what users have liked or disliked in the past, the system will automatically recommend

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 571–582,
Edinburgh, Scotland, UK, July 27–31, 2011.

new items.

Can we provide similar recommendation mechanisms to help users consume large quantities of subjective information? Many commenting environments allow users to mark "like" or "dislike" over existing comments (e.g., Yahoo! News comments, Facebook posts, or review sites that allow helpfulness votes). Can we learn from users' past preferences, so that when a user is reading a new article, we have a system that automatically ranks its comments according to their likelihood of being liked by the user? This can be used directly to create personalized presentation of comments (e.g., into a "like" column and a "dislike" column), as well as enabling down-stream applications such as personalized summarization.

Recommending textual information has recently attracted more attention. So far, the focus has been mainly on recommending news articles (Ahn et al., 2007; Das et al., 2007). Our task differs in several aspects. Intuitively, recommending news articles is largely about identifying the topics of interest to a given user, and it is conceivable that unigram representation of full-length articles can reasonably capture that information. In our case, most comments for an article a user is reading are already of interest to that user topically. Which ones the user ends up liking may depend on several non-topical aspects of the text: whether the user agrees with the viewpoint expressed in the comment, whether the comment is convincing and well-written, etc. Previous work has shown that such analysis can be more difficult than topic-based analysis (Pang and Lee, 2008), and we have the additional challenge that comments are typically much shorter than full-length articles. However, the difficulty in analyzing the textual information in comments can be alleviated by additional contextual information such as author identities. If between a pair of users one consistently likes or dislikes the other, then at least for the heavy users, this authorship information alone could be highly informative. Indeed, previous work in collaborative filtering has usually found no additional gain from leveraging content information when entity-level preference information is abundant.

In this paper, we present a principled way of utilizing multiple sources of information for the task of recommending user comments, which significantly outperforms strong baseline methods, as well as previous methods proposed for text recommendation. While using authorship information alone tends to provide stronger signal than using textual information alone, to our surprise, even for heavy users, adding textual information to the authorship information yields additional improvements.

2 Related Work

There are two main bodies of related work: our problem formulation is closer to collaborative filtering, while the nature of the text we are dealing with has more in common with opinion mining and sentiment analysis.

Our approach is related to a large body of work in collaborative filtering. While a proper survey is not possible here, we describe some of the approaches that are germane. Classical approaches in collaborative filtering are based on item-item/user-user similarity, these are nearest-neighbor methods where the response for a user-item pair is predicted based on a local neighborhood mean (Sarwar et al., 2001; Wang et al., 2006). In general, neighborhoods are defined by measuring similarities between users/items through correlation measures like Pearson, cosine similarities, etc. Better approaches to estimate similarities have also been proposed in Koren (2010). However, modern methods based on matrix factorization have been shown to outperform nearest neighbor methods (Salakhutdinov and Mnih, 2008a,b; Bell et al., 2007). Generalizations of matrix factorization to include both features and past ratings have been proposed (Agarwal and Chen, 2009; Stern et al., 2009). The approach in this paper is an extension where in addition to interactions among users and items (comments in our case), we also consider the authorship information. Three-way interactions were recently studied for personalized tag recommendation (Rendle and Lars, 2010). Their model was based on the sum of two-way interactions, and was trained by using pairwise tag preferences for each (user, item) pair. However, no features were considered, which is an important consideration for us. We show using both text and authorship provides the best performance.

Our work is also related to news personalization that has received increasing attention in the last few

years. For instance, Billsus and Pazanni (2007) describes an approach to build user profile models for adaptive personalization in the context of mobile content access. Their approach is based on a hybrid model that combines content-based approaches with similarity methods used in recommender systems. This is further exemplified in the work by Ahn et al. (2007) where text processing techniques are used to build content profiles for users to recommend personalized news. In our experiments, we show that such approaches are inferior to our method. A content agnostic approach based on collaborative filtering techniques was proposed by Das et al. (2007); cold-start for new items/users was not their focus, but is important for our task — candidate comments for recommendation are often not in training data.

As discussed in Section 1, previous work in opinion mining and sentiment analysis has addressed the information consumption challenge via review summarization. Discussion of early work in that direction can be found in Pang and Lee (2008). In this line of work, opinions for each given aspect are usually summarized as the average sentiment polarity associated with that aspect. Related to that, people have looked into predicting review helpfulness given the textual information in reviews, where helpfulness is either defined as the percentage of users who have voted the review to be helpful (Kim et al., 2006), or labeled by annotators according to a set of criteria (Liu et al., 2007). Our goal differs in that we look for personalized ranking (what a specific user might like) rather than generic quality (what an average user might like). Subsequently, there has been work that tried to predict similarly defined helpfulness scores using meta-information over the reviewer. For instance, whether the author has used his/her true name or where the user is from (Danescu-Niculescu-Mizil et al., 2009), as well as graph structure in the social network between reviewers (Lu et al., 2010). In this work, we simply use author identity to provide more context to the short text; in future work, additional meta-information over users can easily be incorporated via our model.

As discussed in Section 1, whether a rater likes a comment or not may depend on whether they agree with the viewpoint expressed in the text and quality of the text. While previous work has not looked into the reader-comments relationship, there has been related work on identifying political orientations or viewpoints (Lin and Hauptmann, 2006; Lin et al., 2006; Mullen and Malouf, 2006, 2008; Laver et al., 2003); whether a piece of text expresses support or opposition in congressional debates (Thomas et al., 2006) or online debates (Somasundaran and Wiebe, 2009, 2010); as well as identifying contrastive relationship (Kawahara et al., 2010). Note that it is not trivial to use previous work along this line to directly serve as sub-components in our setting. For instance, for work on identifying political orientations or viewpoints, the training data consists of text with the desired labels. In our setting, our labels come in the form of whether users liked or disliked a previous comment. In the simplest case, we might have pair-wise constraints on whether two pieces of text have the same viewpoints (i.e., liked or disliked by the same rater), which would yield a different learning problem akin to the metric learning problem; note, however, the complication that two pieces of text receiving different labels from a given user might not necessarily contain contrasting viewpoints. Consequently, rather than trying to reduce this problem to a set of known text classification tasks, we address this task via a collaborative filtering framework that incorporates textual features.

3 Method

In this section, we describe our model that predicts rater affinity to comments. A key strength of our model is the ability to incorporate rater-comment and rater-author interactions simultaneously in a principled fashion. Our model also provides a seamless mechanism to transition from cold-start (where recommendations need to be made for users or items with no or few past ratings) to warm-start scenarios — with a large amount of data, it fits a per-rater (author) model; with increase in data sparsity, the model applies a small sample size correction through features (in our case, textual features). The exact formula for such corrections in the presence of sparsity is based on parameter estimates that are obtained by applying an EM algorithm to the training data.

3.1 Model

Notation: Let y_{ij} denote the rating that user i, called the *rater*, gives to comment j. Since throughout, we use suffix i to denote a rater and suffix j to denote a comment, we slightly abuse notation and let $\boldsymbol{x}_i$ (of dimension p_u) and $\boldsymbol{x}_j$ (of dimension p_c) denote feature vectors of user i and comment j respectively. For example, $\boldsymbol{x}_i$ can be the bag of words representation (a sparse vector) inferred through text analysis on comments voted positively by user i in the past, and $\boldsymbol{x}_j$ can be the bag of words representation for comment j. We use $a(j)$ to denote the author of comment j, and use μ_{ij} to denote the mean rating by rater i on comment j, i.e., $\mu_{ij} = E(y_{ij})$. Of course it is impossible to estimate μ_{ij} empirically since each user i usually rates a comment j at most once.

Model specification: We work in a generalized linear model framework (McCullagh and Nelder, 1989) that assumes μ_{ij} (or some monotone function h of μ_{ij}) is an additive function of (1) the rater bias α_i of user i since some users may have a tendency of rating comments more positively or negatively than others, (2) popularity β_j of comment j, which could reflect the quality of the comment in this setting, and (3) the author reputation $\gamma_{a(j)}$ of user $a(j)$ since comments by a reputed author may in general get more positive ratings. Thus, the overall bias is $\alpha_i + \beta_j + \gamma_{a(j)}$.

In addition to the bias, we include terms that capture interactions among entities (raters, authors, comments). Indeed, capturing such interactions is a non-trivial part of our modeling procedure. In our approach, we take recourse to *factor* models that have been widely used in collaborative filtering applications in recent times. The basic idea is to attach latent factors to each rater, author and comment. These latent factors are finite dimensional Euclidean vectors that are unknown and estimated from the data. They provide a succinct representation of various *aspects* that are important to explain interaction among entities. In our case, we use the following factors — (a) user factor $\boldsymbol{v}_i$ of dimension $r_v(\geq 1)$ to model rater-author affinity, (b) user factor $\boldsymbol{u}_i$ and comment factor $\boldsymbol{c}_j$ of dimension $r_u(\geq 1)$ to model rater-comment affinity. Intuitively, each could represent viewpoints of users or comments along different dimensions.

i	index for raters
j	index for comments
$a(j)$	author of comment j
y_{ij}	rating given by rater i to comment j
μ_{ij}	mean rating given by rater i to comment j
$\boldsymbol{x}_j$	feature vector of comment j (e.g., textual features in comment j)
$\boldsymbol{x}_i$	feature vector of user i (e.g., comments voted positively by user i)
bias terms:	
α_i	rater bias of user i
β_j	popularity of comment j (e.g., quality of the comment)
$\gamma_{a(j)}$	reputation of the author of comment j
interaction terms:	
$\boldsymbol{v}_i$	user factor for rater-author affinity
$\boldsymbol{u}_i$, $\boldsymbol{c}_j$	factors for rater-comment affinity

Table 1: Table of Notations.

Affinity of rater i to comment j by author $a(j)$ is captured by (1) similarity between viewpoints of users i and $a(j)$, measured by $\boldsymbol{v}_i'\boldsymbol{v}_{a(j)}$; and (2) similarity between the preferences of user i and the perspectives reflected in comment j, measured by $\boldsymbol{u}_i'\boldsymbol{c}_j$. The overall interaction is $\boldsymbol{v}_i'\boldsymbol{v}_{a(j)} + \boldsymbol{u}_i'\boldsymbol{c}_j$. Then, the mean rating μ_{ij}, or more precisely $h(\mu_{ij})$, is modeled as the sum of bias and interaction terms. Mathematically, we assume:

$$\begin{aligned} y_{ij} &\sim N(\mu_{ij},\ \sigma_y^2) \text{ or Bernoulli}(\mu_{ij}) \\ h(\mu_{ij}) &= \alpha_i + \beta_j + \gamma_{a(j)} + \boldsymbol{v}_i'\boldsymbol{v}_{a(j)} + \boldsymbol{u}_i'\boldsymbol{c}_j \end{aligned} \quad (1)$$

For numeric ratings, we use the Gaussian distribution denoted by N(mean,var); for binary ratings, we use the Bernoulli distribution. For Gaussian, $h(\mu_{ij}) = \mu_{ij}$, and for Bernoulli, we assume $h(\mu_{ij}) = \log \frac{\mu_{ij}}{1-\mu_{ij}}$, which is the commonly used logistic transformation.

Table 1 summarizes the notations for easy references. We denote the full model specified above as **vv+uc** since both user-user interaction $\boldsymbol{v}_i'\boldsymbol{v}_{a(j)}$ and user-comment interaction $\boldsymbol{u}_i'\boldsymbol{c}_j$ are modeled at the same time.

Latent factors: A natural approach to estimate latent factors in Equation 1 is through a maximum likelihood estimation (MLE) approach. This does

not work in our scenario since a large fraction of entities have small sample size. For instance, if a comment is rated only by one user and $r_u > 1$, the model is clearly overparametrized and the MLE of the comment factor would tend to learn idiosyncrasies in the training data. Hence, it is imperative to impose constraints on the factors to obtain estimates that generalize well on unseen data. We work in a Bayesian framework where such constraints are imposed through prior distributions. The crucial issue is the selection of appropriate priors. In our scenario, we need priors that provide a good *backoff* estimate when interacting entities have small sample sizes. For instance, to estimate latent factors of a user with little data, we provide a backoff estimate that is obtained by *pooling* data across users with the same user features. We perform such a pooling through regression, the mathematical specification is given below.

$$\begin{aligned} \alpha_i &\sim N(g'\boldsymbol{x}_i,\ \sigma_\alpha^2), & \boldsymbol{u}_i &\sim N(G\boldsymbol{x}_i,\ \sigma_u^2), \\ \beta_j &\sim N(d'\boldsymbol{x}_j,\ \sigma_\beta^2), & \boldsymbol{c}_j &\sim N(D\boldsymbol{x}_j,\ \sigma_c^2), \\ \gamma_{a(j)} &\sim N(0,\ \sigma_\gamma^2), & \boldsymbol{v}_i &\sim N(\mathbf{0},\ \sigma_v^2), \end{aligned}$$

where $g^{p_u \times 1}$ and $d^{p_c \times 1}$ are regression weight vectors, and $G^{r_u \times p_u}$ and $D^{r_u \times p_c}$ are regression weight matrices. These regression weights are learnt from data and provide the backoff estimate. Take the prior distribution of $\boldsymbol{u}_i$ for example. We can rewrite the prior as $\boldsymbol{u}_i = G\boldsymbol{x}_i + \boldsymbol{\delta}_i$, where $\boldsymbol{\delta}_i \sim N(\mathbf{0},\ \sigma_u^2)$. If user i has no rating in the training data, $\boldsymbol{u}_i$ will be predicted as the prior mean (backoff) $G\boldsymbol{x}_i$, a linear projection from the feature vector $\boldsymbol{x}_i$ through matrix G learnt from data. This projection can be thought of as a multivariate linear regression problem with weight matrix G, one weight vector per dimension of $\boldsymbol{u}_i$. However, if user i has many ratings in the training data, we will precisely estimate the per-user *residual* $\boldsymbol{\delta}_i$ that is not captured by the regression $G\boldsymbol{x}_i$. For sample sizes in between these two extremes, the per user residual estimate is "shrunk" toward zero — amount of shrinkage depends on the sample size, past user ratings, variability in ratings on comments rated by the user, and the value of variance components $\sigma_\cdot^2$s.

3.2 Special Cases of Our Model

Our full model (**vv+uc**) includes several existing models explored in collaborative filtering and social networks as special cases.

The matrix factorization model: This model assumes the mean rating of user i on item j is given by $h(\mu_{ij}) = \alpha_i + \beta_j + \boldsymbol{u}_i'\boldsymbol{c}_j$, and the mean of the prior distributions on $\alpha_i, \beta_j, \boldsymbol{u}_i, \boldsymbol{c}_j$ are zero, i.e., $g = d = G = D = \mathbf{0}$. Recent work clearly illustrates that this method obtains better predictive accuracy than classical collaborative filtering techniques based on item-item similarity (Bell et al. (2007)).

The **uc** *model:* This is also a matrix factorization model but with priors based on regressions (i.e., non-zero g, d, G, D). It provides a mechanism to deal with both cold and warm-start scenarios in recommender applications (Agarwal and Chen (2009)).

The **vv** *model:* This model assumes $h(\mu_{ij}) = \alpha_i + \gamma_{a(j)} + \boldsymbol{v}_i'\boldsymbol{v}_{a(j)}$. It was first proposed by Hoff (2005) to model interactions in social networks. The model was fitted to small datasets (at most a few hundred nodes) and the goal was to test certain hypotheses on social behavior, out-of-sample prediction was not considered.

The low-rank **bilinear** *regression model:* Here, $h(\mu_{ij}) = g'\boldsymbol{x}_i + d'\boldsymbol{x}_j + \boldsymbol{x}_i'G'D\boldsymbol{x}_j$. This is a regression model purely based on features with no per-user or per-comment latent factors. In a more general form, $\boldsymbol{x}_i'G'D\boldsymbol{x}_j$ can be written as $\boldsymbol{x}_i'A\boldsymbol{x}_j$, where $A^{p_u \times p_c}$ is the matrix of regression weights (Chu and Park, 2009). However, since $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are typically high dimensional, A can be a large matrix that needs to be learnt from data. To reduce dimensionality, one can decompose A as $A = G'D$, where the number of rows in D and G are small. Thus, instead of learning A, we learn a low-rank approximation of A. This ensures scalability and provides an attractive method to avoid over-fitting.

3.3 Model Fitting

Model fitting for our model is based on the expectation-maximization (EM) algorithm (Dempster et al., 1977). For ease of exposition and space constraints, we only provide a sketch of the algorithm for the Gaussian case, the logistic model can be fitted along the same lines by using a variational approximation (see Agarwal and Chen (2009)).

Let $\boldsymbol{Y} = \{y_{ij}\}$ denote the set of the observed ratings. In the EM parlance, this is "incomplete"

data that gets augmented with the latent factors $\boldsymbol{\Theta} = \{\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{c}_j\}$ to obtain the "complete" data. The goal of the EM algorithm is to find the parameter $\boldsymbol{\eta} = (g, d, G, D, \sigma_\alpha^2, \sigma_\beta^2, \sigma_u^2, \sigma_v^2, \sigma_y^2)$ that maximizes the "incomplete" data likelihood $\Pr(\boldsymbol{Y}|\boldsymbol{\eta}) = \int \Pr(\boldsymbol{Y}, \boldsymbol{\Theta}|\boldsymbol{\eta}) d\boldsymbol{\Theta}$ that is obtained after marginalization (taking expectation) over the distribution of $\boldsymbol{\Theta}$. Since such marginalization is not available in closed form for our model, we use the EM algorithm.

EM algorithm: The complete data log-likelihood $l(\boldsymbol{\eta}; \boldsymbol{Y}, \boldsymbol{\Theta})$ for the full model in the Gaussian case (where $h(\mu_{ij}) = \mu_{ij}$) is given by $l(\boldsymbol{\eta}; \boldsymbol{Y}, \boldsymbol{\Theta}) =$

$$
\begin{aligned}
&-\tfrac{1}{2}\textstyle\sum_{ij}\left((y_{ij}-\mu_{ij})^2/\sigma_y^2 + \log\sigma_y^2\right)\\
&-\tfrac{1}{2}\textstyle\sum_{i}\left((\alpha_i - g'\boldsymbol{x}_i)^2/\sigma_\alpha^2 + \log\sigma_\alpha^2\right)\\
&-\tfrac{1}{2}\textstyle\sum_{j}\left((\beta_j - d'\boldsymbol{x}_j)^2/\sigma_\beta^2 + \log\sigma_\beta^2\right)\\
&-\tfrac{1}{2}\textstyle\sum_{i}\left(\|\boldsymbol{u}_i - G\boldsymbol{x}_i\|^2/\sigma_u^2 + r_u\log\sigma_u^2\right)\\
&-\tfrac{1}{2}\textstyle\sum_{j}\left(\|\boldsymbol{c}_j - D\boldsymbol{x}_j\|^2/\sigma_c^2 + r_u\log\sigma_c^2\right),\\
&-\tfrac{1}{2}\textstyle\sum_{i}\left(\boldsymbol{v}_i'\boldsymbol{v}_i/\sigma_v^2 + r_v\log\sigma_v^2 + \gamma_i^2/\sigma_\gamma^2 + \log\sigma_\gamma^2\right),
\end{aligned}
$$

where r_u is the dimension of factors $\boldsymbol{u}_i$ and $\boldsymbol{c}_j$, and r_v is the dimension of $\boldsymbol{v}_i$. Let $\boldsymbol{\eta}^{(t)}$ denote the estimated parameter setting at the t^{th} iteration. The EM algorithm iterates through the following two steps until convergence.

- **E-step:** Compute $f_t(\boldsymbol{\eta}) = E_{\boldsymbol{\Theta}}[l(\boldsymbol{\eta}; \boldsymbol{Y}, \boldsymbol{\Theta}) \,|\, \boldsymbol{\eta}^{(t)}]$ as a function of $\boldsymbol{\eta}$, where the expectation is taken over the posterior distribution of $(\boldsymbol{\Theta} \,|\, \boldsymbol{\eta}^{(t)}, \boldsymbol{Y})$. Note that here $\boldsymbol{\eta}$ is the input variable of function f_t, but $\boldsymbol{\eta}^{(t)}$ consists of known quantities (determined in the previous iteration).
- **M-step:** Find the $\boldsymbol{\eta}$ that maximizes the expectation computed in the E-step.

$$\boldsymbol{\eta}^{(t+1)} = \arg\max_{\boldsymbol{\eta}} f_t(\boldsymbol{\eta})$$

Since the expectation in the E-step is not available in a closed form, we use a Gibbs sampler to compute the Monte Carlo expectation (Booth and Hobert, 1999). The Gibbs sampler repeats the following procedure L times. It samples α_i, γ_i, β_j, $\boldsymbol{u}_i$, $\boldsymbol{v}_j$, and $\boldsymbol{c}_j$ sequentially one at a time by sampling from the corresponding full conditional distributions. The full conditional distributions are all Gaussian, hence they are easy to sample. Once a Monte Carlo expectation is calculated from the samples, an updated estimate of $\boldsymbol{\eta}$ is obtained in the M-step. The optimization of variance components $\sigma_\cdot^2$s in the M-step is available in closed form, the regression parameters are estimated through off-the-shelf linear regression routines. We note that the posterior distribution of latent factors for known $\boldsymbol{\eta}$ is multi-modal, we have found the Monte Carlo based EM method to outperform other optimization methods like gradient descent in terms of predictive accuracy.

4 Experiments

4.1 Data

We obtained comment rating data between March and May, 2010 from Yahoo! News, with all user IDs anonymized. On this site, users can post comments on news article pages and rate the comments made by others through thumb-up (positive) or thumb-down (negative) votes. Clearly, for articles with very few comments, there is no need to recommend comments. Also, we do not expect deep personalized recommendations for users who have rated very few comments in the past. To focus on instances of interest to us, we restricted ourselves to a subset of the rating data associated with relatively heavy raters. In particular, we formed the experimental dataset by randomly selecting 9,003 raters who provided at least 200 ratings (of which at least 10 were positive and 10 were negative), 189,291 authors who received at least 20 ratings, and 5,088 news articles that received at least 40 comments in the raw dataset during the three-month period. Note that the per entity sample size in the experimental dataset can be smaller than the thresholds specified above. For instance, a rater with more than 200 ratings in the raw dataset can have fewer than 200 in the experimental dataset due to the removal of certain authors or news articles. (See Figure 2 for a distribution of users with different activity levels.) In total, we have 4,440,222 ratings on 1,197,098 comments.

The 5,088 news articles were split into training articles (the earliest 50%), tuning articles (next 5%), and test articles (the last 45%) based on their publication time. The ratings and comments were split into training, tuning, and test sets according to the article they were associated with. All tuning parameters are determined using the tuning set, and performances are reported over the test set. Note that

this training-test split ensures that performance on the test data best simulates our application scenarios. It also creates a completely cold-start situation for comments — no comment in the test set has any past rating in the training set.

4.2 Experimental Setup

Features: All comments were tokenized, lowercased, with stopwords and punctuations removed. We limited the vocabulary to the 10K most frequent tokens in all comments associated with the training articles. (See Section 4.3.3 for a discussion on the effect of the vocabulary size.) For a given comment j, $\boldsymbol{x}_j$ is its bag of words representation, L_2 normalized. For term weighting, we experimented with both presence value and tf-idf weighting. The latter gives slight better performance. Rater feature vector $\boldsymbol{x}_i$ is created by summing over the feature vectors of all comments rated positively by rater i, which is then L_2 normalized.

Methods: We compare the following methods based on our model: The full model **vv+uc**, as well as the three main special cases, **vv**, **uc**, and **bilinear**, as defined in Section 3. The dimensions of $\boldsymbol{v}_i$, $\boldsymbol{u}_i$ and $\boldsymbol{c}_j$ (i.e., r_v and r_u), and the rank of **bilinear** are selected to obtain the best AUC on the tuning set. In our experiments, $r_v = 2, r_u = 3$ and rank of **bilinear** is 3. In addition, we also evaluate the following baseline methods that predict per-user preferences in isolation, primarily based on textual information.

- **Cosine similarity (cos):** $\boldsymbol{x}_i'\boldsymbol{x}_j$. This is simply based on how similar a new comment j is to the comments rater i has liked in the past.
- **Per-user SVM (svm):** For each rater, train a support vector machine (SVM) classifier using only comments ($\boldsymbol{x}_j$) rated by that user.
- **Per-user Naive Bayes (nb):** For each rater, train a Naive Bayes classifier using only comments ($\boldsymbol{x}_j$) rated by that user.[1]

Note that SVMs typically yield the best performance on text classification tasks; a Naive Bayes classifier can be more robust over shorter text spans common in user comments given the high variance. For fair comparisons, for the three baseline methods, we use a simple way of utilizing author information: the feature space is augmented with author IDs and each $\boldsymbol{x}_j$ is augmented with $a(j)$[2]. In Section 4.3, we only report results using the augmented feature vectors since they yield better performance (though the difference is fairly small).

Performance metrics: We use two types of metrics to measure the performance of a method: (1) A global metric based on Receiver Operating Characteristic (ROC) and (2) Precision at rank k (P@k). The former measures the overall correlation of predicted scores for a method with the observed ratings in the test set, while the latter measures the performance of a hypothetical top-k recommendation scenario using the method. To summarize an ROC curve into a single number, we use the Area Under the ROC Curve (AUC). Since random guess yields AUC score of 0.5, regardless of the class distribution, using this measure makes it convenient for us to compare the performance over different subsets of the data (where class distributions could be different). The P@k of a method is computed as follows: (1) For each rater, rank comments that the rater rated in the test set according to the scores predicted by the method, and compute the precision at rank k for that rater; and then (2) average the per-rater precision numbers over all raters. To report P@k, for $k = 5, 10, 20$, we only use raters who have at least 50 ratings in the test set. Statistical significance based on a two-sample t-test across raters is also reported.

4.3 Results

4.3.1 Main comparisons

We first show the ROC curves of different methods on the test set in Figure 1, and the AUCs and precisions in Table 2. Results from significance tests are in Table 3.

First, note that while **svm** significantly outperforms random guesses and **nb**, it is worse than **bilinear**, which is also using (mostly) textual information, but learns the model for all users together,

[1] As we mentioned in Section 4.1, not all users have training data of both classes in the experimental dataset. For **svm** and **nb**, we use the following backoff: for users with training data from only c_i, we predict c_i; for users with no training data at all, we predict the majority class, in this case, the positive class.

[2] We assign weight 1 to $a(j)$, so that the author information have the same impact as the textual features.

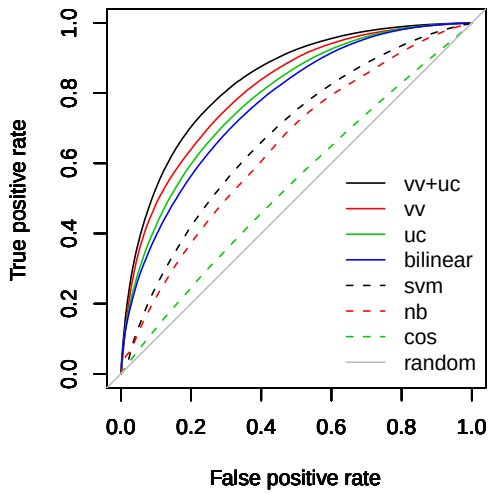


Figure 1: ROC curves of different models

Method	AUC	P@5	P@10	P@20
vv+uc	0.8360	0.9152	0.9079	0.8942
vv	0.8090	0.8810	0.8807	0.8727
uc	0.7857	0.9046	0.8921	0.8694
bilinear	0.7701	0.9028	0.8894	0.8668
svm	0.6768	0.7814	0.7678	0.7497
nb	0.6465	0.7660	0.7486	0.7309
cos	0.5382	0.6834	0.6813	0.6754

Table 2: AUCs and precisions of different models.

rather than in isolation. Next, **uc** outperforms **bilinear** (significantly in AUC, P@10 and P@20), showing per-user and per-comment latent factors help. Note that **vv** outperforms **uc** in ROC, AUC and P@20, but is worse than **uc** in P@5 and P@10; we will take a closer look at this later. Finally, the full model **vv+uc** significantly outperforms both **vv** and **uc**, achieving 0.83 in AUC, and close to 90% in precision at rank 20.

4.3.2 Break-down by user activity level

Next, we investigate model performance in different subsets of the test set. For succinctness, we use AUC as our performance metric. In Figure 2(a), we breakdown model performance by different author activity levels. In Figure 2(b), we breakdown model performance by different voter activity levels. We also generated similar plots with the y-axis replaced by P@5, P@10 and P@20, and observed the same trend except that **vv** starts to outperform **uc** at different user activity thresholds for different metrics.

Comparison	Metrics	p-value
vv+uc $>$ **vv**	All	$< 10^{-7}$
vv+uc $>$ **uc**	All	$< 10^{-20}$
uc $>$ **bilinear**	All except P@5	< 0.006
bilinear $>$ **svm**	All	$< 10^{-20}$
vv $>$ **svm**	All	$< 10^{-20}$
svm $>$ **nb**	All	$< 10^{-8}$
nb $>$ **cos**	All	$< 10^{-20}$

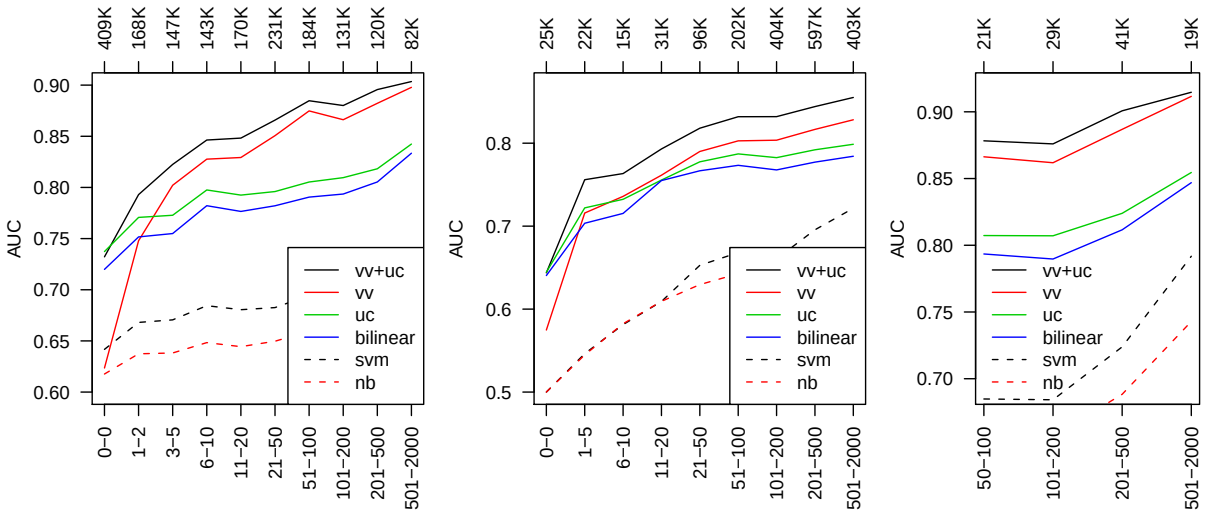
Table 3: Paired t-test results. Note that **uc** is better than **bilinear** in P@5, but not significant. The orders of **uc** and **vv** are not consistent across different metrics.

Not surprisingly, **vv** performs poorly for raters or authors with no ratings observed in the training data. However, once we have a small amount of ratings, it starts to outperform **uc**, even though intuitively, the textual information in the comment should be more informative than the authorship information alone. Using paired t-tests with significance level 0.05, we report when **vv** starts to significantly outperform **uc** in the following table, which is interpreted as follows — **vv** is not significantly worse than **uc** in metric M if the author of a test comment received at least N_{eq} ratings in the training set, and **vv** significantly outperforms **uc** in metric M if the author received at least N_{gt} ratings in the training set.

Metric M	P@5	P@10	P@20	AUC
# Ratings N_{eq}	50	5	5	5
N_{gt}	1000	50	5	5

Recall that our training/test split is by article. Since we have never observed a rater's preference over the test articles before, it is rather surprising that author information alone can yield 0.8 in AUC score, even for very light authors who have received between 3 and 5 votes in total in the training data. This suggests that users' viewpoints are quite consistent: a large portion of the ratings can be adequately explained by the pair of user identities. One interesting observation is that the number of ratings required for **vv** to outperform **uc** in P@5 is quite high. This suggests that to obtain high precision at the top of a recommended list, comment features are important.

Nonetheless, modeling textual information in addition to author information provides additional improvements. Based on paired t-tests with signifi-



(a) AUC by author activity levels (b) AUC by rater activity levels (c) AUC by user activity levels

Figure 2: AUC of different models as a function of the activity level of authors or raters. The x-axis (bottom) has the form m-n, meaning the subset of the test data in which the number of ratings that each author received (as in (a)) or each rater gave (as in (b)) in the training set is between m and n. In (c), we select both authors and raters based on the m-n criterion. The x-axis (top) denotes the number of ratings in the subset

cance level 0.05, **vv+uc** significantly outperforms **vv** in all metrics if the author received < 500 ratings in the training set. Except for the very heavy authors, even for cases where both raters and authors are heavy users (Figure 2(c)), adding the comment feature information still yields additional improvement over the already impressive performance of using **vv** alone. In spite of the simple representation we adopted for the textual information, the full model is still capable of accounting for part of the residual errors from **vv** model (that uses authorship information alone) by using comment features — what was actually written does matter.

Finally, if we breakdown the comparison between **vv+uc** and **uc** for different user activity levels, **vv+uc** significantly outperforms **uc** (with level 0.05) in all metrics if the author received at least 5 ratings in the training set.

4.3.3 Analysis of textual features

Recall that we limited the vocabulary size to the 10K most frequent terms for efficiency reasons. Is this limitation likely to affect our model performance significantly? We examined the effect of different numbers of features. In the following table, #features = n means that both $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are bags of n words[3]. Since the **vv** model does not utilize rater or comment features, we examine AUC of the **uc** model.

#features	1K	3K	5K	10K
AUC	0.7713	0.7855	0.7872	0.7876

As can be seen, the performance improvement is in the 4th decimal place when we increase from 5K features to 10K features. Thus, we do not further increase the number of features in our experiments.

Note that our full model does not require rater features and comment features to be in the same feature space. Each is projected into the hidden "viewpoint" space, via G and D, separately. For simplicity and easy comparison to other methods, we used all comments liked by a rater in the past to build the feature vector of the rater. But since the full model already has information of the textual content of comments from the comment features, and which comments were liked by the users from the ratings, rater features constructed this way do not provide any new information. Indeed, if we model $\boldsymbol{u}_i \sim N(1, \sigma_u^2)$, instead of $\boldsymbol{u}_i \sim N(G\boldsymbol{x}_i, \sigma_u^2)$, this omission of x_i does not hurt the performance of the model. In future work, other meta-information about the rater

[3]Note that we used n most useful features in each case.

can easily be incorporated into $\boldsymbol{x}_i$ to enrich rater representation.

Recall that comment features $\boldsymbol{x}_j$ were projected to comment factors $\boldsymbol{c}_j$ via D. We envisioned that the comment factors could be representing viewpoints. Does our model conform to this intuition? Let's consider the simplest case, where we restrict $\boldsymbol{u}_i$ and $\boldsymbol{c}_j$ to be one-dimensional vectors. In this case, each can be represented by scalars u_i and c_j. If u_i and c_j are of the same sign, then the rater is likely to like the comment. Words assigned high positive weights or low negative weights via D will have significant contributions to the overall sign of c_j. Now if we examine such words, will we see any meaningful differences in the underlying viewpoints of these two groups of words?

To address this question qualitatively, we manually sampled words with heavy weights, focusing on politics-related ones (so that viewpoints are likely to be polarized and easier to interpret). At one extreme, we observe words like *repukes, repugs*, which seemed to be derogatory mentions of Republications, and likely to represent an anti-Republication point of view. At the other end, we observe terms like *libtards, nobama, obummer*. While terms like *nobama* may appear to be typos at first sight, a quick search online reveals that these are at least intentional typos expressing anti-Obama sentiments, which clearly represents an opposite underlying perspective from terms like *repukes*.

These examples also illustrate the importance to learn directly from the data of interest to us. Such indicative words would never have appeared in more formal writings. While we do not have direct labels for perspectives, our model seems to be capturing the underlying perspectives (as much as a unigram-based model could) by learning from user preference labels across different users. This allows us to learn the text features most relevant to our dataset, which is particularly important given the time-sensitive and ever-evolving nature of news-related comments.

5 Conclusions

In this paper, we promote personalized recommendation as a novel way of helping users to consume large quantities of subjective information. We propose using a principled way of incorporating both rater-comment and rater-author interactions simultaneously. Our full model significantly outperforms strong baseline methods, as well as previous methods proposed for text recommendation. In particular, learning weights over textual features across all users outperforms learning for each user individually, which holds true even for heavy raters. Furthermore, while using authorship information alone provides stronger signal than using textual information alone, to our surprise, even for heavy users, adding textual information yields additional improvements.

It is difficult to comprehensively capture user affinity to comments using a finite number of ratings observed during a certain time interval. News and comments on news articles are dynamic in nature, novel aspects may emerge over time. To capture such dynamic behavior, comment factors have to be allowed to evolve over time and such an evolution would also necessitate the re-estimation of user factors. Incorporating such temporal dynamics into our modeling framework is a challenging research problem and requires significant elaboration of our current approach.

Acknowledgments

We thank the anonymous reviewers for useful suggestions.

References

D. Agarwal and B.C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM, 2009.

Jae-Wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Y. Syn. Open user profiles for adaptive news systems: help or harm? In *Proceedings of the 16th international conference on World Wide Web (WWW)*, 2007.

Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD*, 2007.

D. Billsus and M. Pazanni. Adaptive news access. Springer, Berlin, 2007.

J.G. Booth and J.P Hobert. Maximizing generalized linear mixed model likelihoods with an automated

monte carlo EM algorithm. *J.R.Statist. Soc. B*, 1999.

Wei Chu and Seung T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th international conference on World Wide Web (WWW)*, 2009.

Cristian Danescu-Niculescu-Mizil, Gueorgi Kossinets, Jon Kleinberg, and Lillian Lee. How opinions are received by online communities: A case study on Amazon.com helpfulness votes. In *Proceedings of WWW*, pages 141–150, 2009.

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web (WWW)*, 2007.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

Peter D. Hoff. Bilinear mixed-effects models for dyadic data. *Journal of the American Statistical Association*, 100(469):286–295, 2005.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 168–177, 2004.

Daisuke Kawahara, Kentaro Inui, and Sadao Kurohashi. Identifying contradictory and contrastive relations between statements to outline web information on a given topic. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING): Posters*, 2010.

Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 423–430, Sydney, Australia, July 2006. Association for Computational Linguistics.

Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4 (1):1–24, 2010. ISSN 1556-4681.

Michael Laver, Kenneth Benoit, and John Garry. Extracting policy positions from political texts using words as data. *American Political Science Review*, 97(2):311–331, 2003.

Wei-Hao Lin and Alexander Hauptmann. Are these documents written from different perspectives? A test of different perspectives based on sta tistical distribution divergence. In *Proceedings of the International Conference on Computational Linguistics (COLING)/Proceedings of the Association for Computational Linguistics (ACL)*, pages 1057–1064, Sydney, Australia, July 2006. Association for Computational Linguistics.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. Which side are you on? identifying perspectives at the document and sentence levels. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, 2006.

Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. Low-quality product review detection in opinion summarization. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 334–342, 2007. Poster paper.

Yue Lu, Panayiotis Tsaparas, Alexandros Ntoulas, and Livia Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th International World Wide Web Conference (WWW)*, 2010.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman & Hall/CRC, 1989.

Tony Mullen and Robert Malouf. A preliminary investigation into sentiment analysis of informal political discourse. In *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, pages 159–162, 2006.

Tony Mullen and Robert Malouf. Taking sides: User classification for informal online political discourse. *Internet Research*, 18:177–190, 2008.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Pro cessing (HLT/EMNLP)*, 2005.

Steffen Rendle and Schmidt-Thie Lars. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 81–90, New York, NY, USA, 2010. ACM.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008a.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20:1257–1264, 2008b.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

Benjamin Snyder and Regina Barzilay. Multiple aspect ranking using the Good Grief algorithm. In *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT- NAACL)*, pages 300–307, 2007.

Swapna Somasundaran and Janyce Wiebe. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009.

Swapna Somasundaran and Janyce Wiebe. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, 2010.

David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World Wide Web (WWW)*, 2009.

Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 327–335, 2006.

Ivan Titov and Ryan McDonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2008.

Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA, 2006. ACM.

Data-Driven Response Generation in Social Media

Alan Ritter
Computer Sci. & Eng.
University of Washington
Seattle, WA 98195
aritter@cs.washington.edu

Colin Cherry
National Research Council Canada
Ottawa, Ontario, K1A 0R6
Colin.Cherry@nrc-cnrc.gc.ca

William B. Dolan
Microsoft Research
Redmond, WA 98052
billdol@microsoft.com

Abstract

We present a data-driven approach to generating responses to Twitter status posts, based on phrase-based Statistical Machine Translation. We find that mapping conversational stimuli onto responses is more difficult than translating between languages, due to the wider range of possible responses, the larger fraction of unaligned words/phrases, and the presence of large phrase pairs whose alignment cannot be further decomposed. After addressing these challenges, we compare approaches based on SMT and Information Retrieval in a human evaluation. We show that SMT outperforms IR on this task, and its output is preferred over actual human responses in 15% of cases. As far as we are aware, this is the first work to investigate the use of phrase-based SMT to directly translate a linguistic stimulus into an appropriate response.

1 Introduction

Recently there has been an explosion in the number of people having informal, public conversations on social media websites such as Facebook and Twitter. This presents a unique opportunity to build collections of naturally occurring conversations that are orders of magnitude larger than those previously available. These corpora, in turn, present new opportunities to apply data-driven techniques to conversational tasks.

We investigate the problem of **response generation**: given a conversational stimulus, generate an appropriate response. Specifically, we employ a large corpus of status-response pairs found on Twitter to create a system that responds to Twitter status posts. Note that we make no mention of context, intent or dialogue state; our goal is to generate any response that fits the provided stimulus; however, we do so without employing rules or templates, with the hope of creating a system that is both flexible and extensible when operating in an open domain.

Success in open domain response generation could be immediately useful to social media platforms, providing a list of suggested responses to a target status, or providing conversation-aware autocomplete for responses in progress. These features are especially important on hand-held devices (Hasselgren et al., 2003). Response generation should also be beneficial in building "chatterbots" (Weizenbaum, 1966) for entertainment purposes or companionship (Wilks, 2006). However, we are most excited by the future potential of data-driven response generation when used inside larger dialogue systems, where direct consideration of the user's utterance could be combined with dialogue state (Wong and Mooney, 2007; Langner et al., 2010) to generate locally coherent, purposeful dialogue.

In this work, we investigate statistical machine translation as an approach for response generation. We are motivated by the following observation: In naturally occurring discourse, there is often a strong structural relationship between adjacent utterances (Hobbs, 1985). For example, consider the stimulus-response pair from the data:

> **Stimulus**: I'm slowly making this soup ..... and it smells gorgeous!

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 583–593,
Edinburgh, Scotland, UK, July 27–31, 2011.

Response: I'll bet it looks delicious too! Haha

Here "it" in the response refers to "this soup" in the status by co-reference; however, there is also a more subtle relationship between the "smells" and "looks", as well as "gorgeous" and "delicious". Parallelisms such as these are frequent in naturally occurring conversations, leading us to ask whether it might be possible to *translate* a stimulus into an appropriate response. We apply SMT to this problem, treating Twitter as our parallel corpus, with status posts as our source language and their responses as our target language. However, the established SMT pipeline cannot simply be applied out of the box.

We identify two key challenges in adapting SMT to the response generation task. First, unlike bilingual text, stimulus-response pairs are not semantically equivalent, leading to a wider range of possible responses for a given stimulus phrase. Furthermore, both sides of our parallel text are written in the same language. Thus, the most strongly associated word or phrase pairs found by off-the-shelf word alignment and phrase extraction tools are identical pairs. We address this issue with constraints and features to limit lexical overlap. Secondly, in stimulus-response pairs, there are far more unaligned words than in bilingual pairs; it is often the case that large portions of the stimulus are not referenced in the response and vice versa. Also, there are more large phrase-pairs that cannot be easily decomposed (for example see figure 2). These difficult cases confuse the IBM word alignment models. Instead of relying on these alignments to extract phrase-pairs, we consider all possible phrase-pairs in our parallel text, and apply an association-based filter.

We compare our approach to response generation against two Information Retrieval or nearest neighbour approaches, which use the input stimulus to select a response directly from the training data. We analyze the advantages and disadvantages of each approach, and perform an evaluation using human annotators from Amazon's Mechanical Turk. Along the way, we investigate the utility of SMT's BLEU evaluation metric when applied to this domain. We show that SMT-based solutions outperform IR-based solutions, and are chosen over actual human responses in our data in 15% of cases. As far as we are aware, this is the first work to investigate the feasibility of SMT's application to generating responses to open-domain linguistic stimuli.

2 Related Work

There has been a long history of "chatterbots" (Weizenbaum, 1966; Isbell et al., 2000; Shaikh et al., 2010), which attempt to engage users, typically leading the topic of conversation. They usually limit interactions to a specific scenario (e.g. a Rogerian psychotherapist), and use a set of template rules for generating responses. In contrast, we focus on the simpler task of generating an appropriate response to a single utterance. We leverage large amounts of conversational training data to scale to our Social Media domain, where conversations can be on just about any topic.

Additionally, there has been work on generating more natural utterances in goal-directed dialogue systems (Ratnaparkhi, 2000; Rambow et al., 2001). Currently, most dialogue systems rely on either canned responses or templates for generation, which can result in utterances which sound very unnatural in context (Chambers and Allen, 2004). Recent work has investigated the use of SMT in translating internal dialogue state into natural language (Langner et al., 2010). In addition to dialogue state, we believe it may be beneficial to consider the user's utterance when generating responses in order to generate locally coherent discourse (Barzilay and Lapata, 2005). Data-driven generation based on users' utterances might also be a useful way to fill in knowledge gaps in the system (Galley et al., 2001; Knight and Hatzivassiloglou, 1995).

Statistical machine translation has been applied to a smörgåsbord of NLP problems, including question answering (Echihabi and Marcu, 2003), semantic parsing and generation (Wong and Mooney, 2006; Wong and Mooney, 2007), summarization (Daumé III and Marcu, 2009), generating bid-phrases in online advertising (Ravi et al., 2010), spelling correction (Sun et al., 2010), paraphrase (Dolan et al., 2004; Quirk et al., 2004) and query expansion (Riezler et al., 2007). Most relevant to our efforts is the work by Soricut and Marcu (2006), who applied the IBM word alignment models to a discourse ordering task, exploiting the same intuition investigated

in this paper: certain words (or phrases) tend to trigger the usage of other words in subsequent discourse units. As far as we are aware, ours is the first work to explore the use of phrase-based translation in generating responses to open-domain linguistic stimuli, although the analogy between translation and dialogue has been drawn (Leuski and Traum, 2010).

3 Data

For learning response-generation models, we use a corpus of roughly 1.3 million conversations scraped from Twitter (Ritter et al., 2010; Danescu-Niculescu-Mizil et al., 2011). Twitter conversations don't occur in real-time as in IRC; rather as in email, users typically take turns responding to each other. Twitter's 140 character limit, however, keeps conversations chat-like. In addition, the Twitter API maintains a reference from each reply to the post it responds to, so unlike IRC, there is no need for conversation disentanglement (Elsner and Charniak, 2008; Wang and Oard, 2009). The first message of a conversation is typically unique, not directed at any particular user but instead broadcast to the author's followers (a status message). For the purposes of this paper, we limit the data set to only the first two utterances from each conversation. As a result of this constraint, any system trained with this data will be specialized for responding to Twitter status posts.

4 Response Generation as Translation

When applied to conversations, SMT models the probability of a response r given the input status-post s using a log-linear combination of feature functions. Most prominent among these features are the conditional phrase-translation probabilities in both directions, $P(s|r)$ and $P(r|s)$, which ensure r is an appropriate response to s, and the *language model* $P(r)$, which ensures r is a well-formed response. As in translation, the response models are estimated from counts of phrase pairs observed in the training bitext, and the language model is built using n-gram statistics from a large set of observed responses. To find the best response to a given input status-post, we employ the Moses phrase-based decoder (Koehn et al., 2007), which conducts a beam search for the best response given the input, according to the log-linear model.

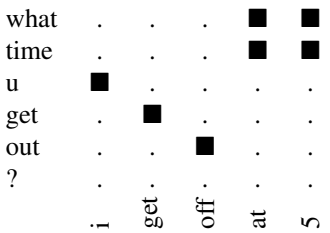


Figure 1: Example from the data where word alignment is easy. There is a clear correspondence between words in the status and the response.

4.1 Challenge: Lexical Repetition

When applied directly to conversation data, off-the-shelf MT systems simply learn to parrot back the input, sometimes with slight modification. For example, directly applying Moses with default settings to the conversation data produces a system which yields the following (typical) output on the above example:

> **Stimulus**: I'm slowly making this soup ..... and it smells gorgeous!
>
> **Response:** i'm slowly making this soup ..... and you smell gorgeous!

This "paraphrasing" phenomenon occurs because identical word pairs are frequently observed together in the training data. Because there is a wide range of acceptable responses to any status, these identical pairs have the strongest associations in the data, and therefore dominate the phrase table. In order to discourage lexically similar translations, we filter out all phrase-pairs where one phrase is a substring of the other, and introduce a novel feature to penalize lexical similarity:

$$\phi_{\text{lex}}(s,t) = J(s,t)$$

Where $J(s,t)$ is the Jaccard similarity between the set of words in s and t.

4.2 Challenge: Word Alignment

Alignment is more difficult in conversational data than bilingual data (Brown et al., 1990), or textual entailment data (Brockett, 2006; MacCartney et al., 2008). In conversational data, there are some cases in which there is a decomposable alignment between

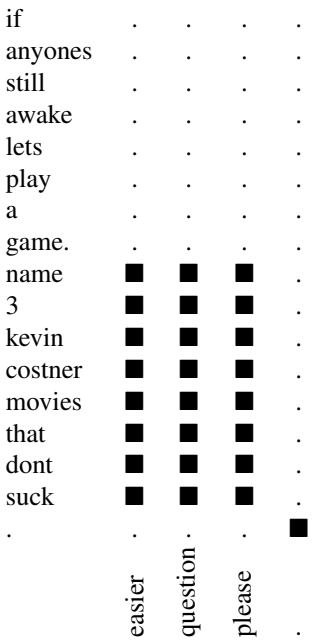


Figure 2: Example from the data where word alignment is difficult (requires alignment between large phrases in the status and response).

words, as seen in figure 1, and some difficult cases where alignment between large phrases is required, for example figure 2. These difficult sentence pairs confuse the IBM word alignment models which have no way to distinguish between the easy and hard cases.

We aligned words in our parallel data using the widely used tool GIZA++ (Och and Ney, 2003); however, the standard growing heuristic resulted in very noisy alignments. Precision could be improved considerably by using the intersection of GIZA++ trained in two directions ($s \rightarrow r$, and $r \rightarrow s$), but the alignment also became extremely sparse. The average number of alignments-per status/response pair in our data was only 1.7, as compared to a dataset of aligned French-English sentence pairs (the WMT 08 news commentary data) where the average number of intersection alignments is 14.

Direct Phrase Pair Extraction

Because word alignment in status/response pairs is a difficult problem, instead of relying on local alignments for extracting phrase pairs, we exploit information from all occurrences of the pair in determining whether its phrases form a valid mapping.

$C(s,t)$	$C(s,\neg t)$	$C(s)$
$C(\neg s,t)$	$C(\neg s,\neg t)$	$N-C(s)$
$C(t)$	$N-C(t)$	N

Figure 3: Contingency table for phrase pair (s,t). Fisher's Exact Test estimates the probability of seeing this event, or one more extreme assuming s and t are independent.

We consider all possible phrase-pairs in the training data,[1] then use Fisher's Exact Test to filter out pairs with low correlation (Johnson et al., 2007). Given a source and target phrase s and t, we consider the contingency table illustrated in figure 3, which includes co-occurrence counts for s and t, the number of sentence-pairs containing s, but not t and vice versa, in addition to the number of pairs containing neither s nor t. Fisher's Exact Test provides us with an estimate of the probability of observing this table, or one more extreme, assuming s and t are independent; in other words it gives us a measure of how strongly associated they are. In contrast to statistical tests such as χ^2, or the G^2 Log Likelihood Ratio, Fisher's Exact Test produces accurate p-values even when the expected counts are small (as is extremely common in our case).

In Fisher's Exact Test, the hypergeometric probability distribution is used to compute the exact probability of a particular joint frequency assuming a model of independence:

$$\frac{C(s)!C(\neg s)!C(t)!C(\neg t)!}{N!C(s,t)!C(\neg s,t)!C(s,\neg t)!C(\neg s,\neg t)!}$$

The statistic is computed by summing the probability for the joint frequency in Table 3, and every more extreme joint frequency consistent with the marginal frequencies. Moore (2004) illustrates several tricks which make this computation feasible in practice.

We found that this approach generates phrase-table entries which appear quite reasonable upon manual inspection. The top 20 phrase-pairs (after filtering out identical source/target phrases, substrings,

[1] We define a possible phrase-pair as any pair of phrases found in a sentence-pair from our training corpus, where both phrases consist of 4 tokens or fewer. The total number of phrase pairs in a sentence pair (s,r) is $O(|s| \times |r|)$.

Source	Target
rt [*retweet*]	thanks for the
potter	harry
ice	cream
how are you	you ?
good	morning
chuck	norris
watching	movie
i miss	miss you too
are you	i 'm
my birthday	happy birthday
wish me luck	good luck
how was	it was
miss you	i miss
swine	flu
i love you	love you too
how are	are you ?
did you	i did
jackson	michael
how are you	i 'm good
michael	mj

Table 1: Top 20 Phrase Pairs ranked by the Fisher Exact Test statistic. Slight variations (substrings or symmetric pairs) were removed to show more variety. See the supplementary materials for the top 10k (unfiltered) pairs.

and symmetric pairs) are listed in Table 1.[2] Our experiments in §6 show that using the phrase table produced by Fisher's Exact Test outperforms one generated based on the poor quality IBM word alignments.

4.3 System Details

For the phrase-table used in the experiments (§6) we used the 5M phrases with highest association according the Fisher Exact Test statistic.[3] To build the language model, we used all of the 1.3M responses from the training data, along with roughly 1M replies collected using Twitter's streaming API.

We do not use any form of SMT reordering model, as the position of the phrase in the response does not seem to be very correlated with the corresponding position in the status. Instead we let the language model drive reordering.

We used the default feature weights provided by Moses.[4] Because automatic evaluation of response generation is an open problem, we avoided the use of discriminative training algorithms such as Minimum Error-Rate Training (Och, 2003).

5 Information Retrieval

One straightforward data-driven approach to response generation is nearest neighbour, or information retrieval. This general approach has been applied previously by several authors (Isbell et al., 2000; Swanson and Gordon, 2008; Jafarpour and Burges, 2010), and is used as a point of comparison in our experiments. Given a novel status s and a training corpus of status/response pairs, two retrieval strategies can be used to return a best response r':

IR-STATUS $[r_{\text{argmax}_i\, sim(s,s_i)}]$ Retrieve the response r_i whose associated status message s_i is most similar to the user's input s.

IR-RESPONSE $[r_{\text{argmax}_i\, sim(s,r_i)}]$ Retrieve the response r_i which has highest similarity when directly compared to s.

At first glance, IR-STATUS may appear to be the most promising option; intuitively, if an input status is very similar to a training status, we might expect the corresponding training response to pair well with the input. However, as we describe in §6, it turns out that directly retrieving the most similar response (IR-RESPONSE) tends to return acceptable replies more reliably, as judged by human annotators. To implement our two IR response generators, we rely on the default similarity measure implemented in the Lucene[5] Information Retrieval Library, which is an IDF-weighted Vector-Space similarity.

6 Experiments

In order to compare various approaches to automated response generation, we used human evalu-

[2] See the supplementary materials for the top 10k (unfiltered) phrase pairs.

[3] Note that this includes an arbitrary subset of the (1,1,1) pairs (phrase pairs where both phrases were only observed once in the data). Excluding these (1,1,1) pairs yields a rather small phrase table, 201K phrase-pairs after filtering, while including all of them led to a table which was too large for the memory of the machine used to conduct the experiments.

[4] The language model weight was set to 0.5, the translation model weights in both directions were both set to 0.2, the lexical similarity weight was set to -0.2.

[5] `http://lucene.apache.org/`

ators from Amazon's Mechanical Turk (Snow et al., 2008). Human evaluation also provides us with data for a preliminary investigation into the feasibility of automatic evaluation metrics. While automated evaluation has been investigated in the area of spoken dialogue systems (Jung et al., 2009), it is unclear how well it will correlate with human judgment in open-domain conversations where the range of possible responses is very large.

6.1 Experimental Conditions

We performed pairwise comparisons of several response-generation systems. Similar work on evaluating MT output (Bloodgood and Callison-Burch, 2010) has asked Turkers to rank more than two choices, but in order to keep our evaluation as straightforward as possible, we limited our experiments to pairwise comparisons.

For each experiment comparing 2 systems (*a* and *b*), we built a test set by selecting a random sample of 200 tweets which had received responses, and which had a length between 4 and 20 words. These tweets were selected from conversations collected from a later, non-overlapping time-period from those used in training. Each experiment used a different random sample of 200 tweets. For each of the 200 statuses, we generated a response using method *a* and *b*, then showed the status and both responses to the Turkers, asking them to choose the best response. The order of the systems used to generate a response was randomized, and each of the 200 HITs was submitted to 3 different Turkers. Turkers were paid 1¢ per judgment.

The Turkers were instructed that an appropriate response should be on the same topic as the status, and should also "make sense" in response to it. While this is an inherently subjective task, from inspecting the results, we found Turkers to be quite competent in judging between two responses.

The systems used in these pairwise comparisons are summarized in table 2, and example output generated by each system is presented in Table 3.

System Name	Description
RND-BASELINE	Picks randomly from the set of responses which are observed at least twice in the training data. The assumption is these are likely very general responses
IR-STATUS	$r_{\text{argmax}_i\, sim(s,s_i)}$ as described in §5
IR-RESPONSE	$r_{\text{argmax}_i\, sim(s,r_i)}$ as described in §5
MT-CHAT	Phrase-based translation system as described in §4
MT-BASELINE	Exactly the same as MT-CHAT, except using a phrase table extracted based on word alignments from GIZA++
HUMAN	Actual responses from the test data.

Table 2: Summary of systems compared experimentally

6.2 Results

The results of the experiments are summarized in Table 4. For each experiment we show the fraction of HITs where the majority of annotators agreed system *a* was better. We also show the p-value from an exact Binomial significance test; note that all differences are significant with above 95% confidence. Table 4 also shows the S coefficient (Bennett et al., 1954) as a measure of agreement between annotators for each experiment. S is equivalent the commonly used Kappa, except it assumes the uniform distribution when computing expected or chance agreement (Artstein and Poesio, 2008). The uniform distribution is appropriate in our setup, since annotators are not told which system generated each output, and the order of choices is randomized. If annotators were indeed choosing randomly, they would be expected to agree in 50% of cases. In most comparisons the value of S is in the range of 0.2 to 0.4, which can be considered "Fair agreement" (Landis and Koch, 1977).

Note that agreement between annotators is lower than typically reported in corpus annotation tasks. When annotating which of two automatically generated outputs is better, there is not always a clear answer; both responses might be good or bad. We can expect strong agreement only in cases where one response is clearly better. Strong agreement is not required, however, as we are using many annotations to compare each pair of systems, and the human judgments are not intended to be used as training data. Similar agreement was reported in an evaluation of automatically generated MT output as part

Status	MT-CHAT	MT-BASELINE	IR-STATUS	HUMAN	RND-BASELINE	IR-RESPONSE
riding to the arc on my bike. Hopefully I won't melt into the pavement.	good luck on your bike ride arc	i ca n't wait for it ! i 'm arc is a year .	grab some refreshing drinks ... and a towel ... lol ... then eat ice cream and suika later ...	make sure to ride in the middle of the street	lmfaoooo !	u sure the cup wo n't melt ?
I really hate watching movies in the living room.	me too , what movies are you doing in room ?	me too ! i love those ?) in my mouth .	:(cheer up ! you need to invest in one of those turbo fans	Why? no comfy spots or just too open to public	tas fazendo oq ?	someone was watching in the living room kasi e . :) i was too lazy to go up to my room . :))
Hello Did I miss David Getting his teen choice award? Or did they not show it :(	hello miss ? * he was getting his will probably win .	it 's just like what you just did the exact date hello ?	it did n't show yet -ha- -lol-	nothing yet...he presented though he is so darn cute	you wish .	are they gon na show teen choice awards again ? i missed it
I can think the impossible is possible at any given moment.	i think you can do it , anything is given impossible dreams really specs	i ca n't wait to see if there is no point in the end but if you do a few hours ...	i hope you use those powers for evil . i mean good ! good . that 's what i meant .	for sure Eric, and that makes it not 'impossible' any longer.	no -user- ?	nobody with whom i might apply the blue cobalt ' rabbit ' at the moment . given the economy , perhaps -url- (any of em)
I want to enter my parents house into a home makeover contest. Any help out there?	me too ! thanks for the win , parents are so smart , just got home makeover	oh man , i do n't want to come back to my mom is getting that	i put troy on the potty everyday at lunch and b4 u know it he was going on his own ... only took a week .	check TLC I'm pretty sure it was them who were recently posting about looking for houses to be nominated!	good job	you want to do laundry with me at my parents house after i get off ? maybe get a free meal out of it ? !

Table 3: Example responses from each system. We tried to pick examples where most (or all) systems generate reasonable responses for illustration purposes.

System A	**System B**	**Fraction A**	**p-value**	**Agreement** (S)	**System A BLEU**	**System B BLEU**
MT-CHAT*	IR-STATUS	0.645	5.0e-05	0.347	1.15	0.57
MT-CHAT*	IR-RESPONSE	0.593	1.0e-02	0.333	0.84	1.53
IR-STATUS	IR-RESPONSE*	0.422	3.3e-02	0.330	0.40	1.59
MT-CHAT*	MT-BASELINE	0.577	3.8e-02	0.160	1.23	1.14
MT-CHAT	HUMAN*	0.145	2.2e-16	0.433	N/A	N/A
MT-CHAT*	RND-BASELINE	0.880	2.2e-16	0.383	1.17	0.10

Table 4: Results of pairwise comparisons between various response-generation methods. Each row presents a comparison between systems *a* and *b* on 200 randomly selected tweets. The column **Fraction A** lists the fraction of HITs where the majority of annotators agreed **System A**'s response was better. The winning system is indicated with an asterisk*. All differences are significant.

of the WMT09 shared tasks (Callison-Burch et al., 2009).[6]

The results of the paired evaluations provide a clear ordering on the automatic systems: IR-STATUS is outperformed by IR-RESPONSE, which is in turn outperformed by MT-CHAT. These results are somewhat surprising. We had expected that matching status to status would create a more natural and effective IR system, but in practice, it appears that the additional level of indirection employed by IR-STATUS created only more opportunity for confusion and error. Also, we did not necessarily expect MT-CHAT's output to be preferred by human annotators: the SMT system is the only one that generates a completely novel response, and is therefore the system most likely to make fluency errors. We had expected human annotators to pick up on these fluency errors, giving the the advantage to the IR systems. However, it appears that MT-CHAT's ability to tailor its response to the status on a fine-grained scale overcame the disadvantage of occasionally introducing fluency errors.[7]

Given MT-CHAT's success over the IR systems, we conducted further experiments to validate its output. In order to test how close MT-CHAT's responses come to human-level abilities, we compared its output to actual human responses from our dataset. In some cases the human responses change the topic of conversation, and completely ignore the initial status. For instance, one frequent type of response we noticed in the data was a greeting: "How have you been? I haven't talked to you in a while." For the purposes of this evaluation, we manually filtered out cases where the human response was completely off-topic from the status, selecting 200 pairs at random that met our criteria and using the actual responses as the HUMAN output.

When compared to the actual human-generated response, MT-CHAT loses. However, its output is preferred over the human responses 15% of the time, a fact that is particularly surprising given the very small – by MT standards – amount of data used to train the model. A few examples where MT-CHAT's output were selected over the human response are listed in Table 5.

We also evaluated the effect of filtering all possible phrase pairs using Fisher's Exact Test, which we did instead of conducting phrase extraction according to the very noisy word alignments. We altered our MT-CHAT system to use the standard Moses phrase-extraction pipeline, creating the system denoted as MT-BASELINE. We compared this to the complete MT-CHAT system. Note that both systems account for lexical repetition as described in §4.1. MT-CHAT's output is preferred 58% of the time over MT-BASELINE, indicating that direct phrase extraction is useful in this conversational setting.

Finally, as an additional baseline, we compared MT-CHAT's output to random responses selected from those observed 2 or more times in the training data. One might argue that short, common responses are very general, and that a reply like "lol" could be considered a good response to almost any status. However, the human evaluation shows a clear preference for MT-CHAT's output: raters favour responses that are tailored to the stimulus.

6.3 Automatic Evaluation

The field of SMT has benefited greatly from the existence of an automatic evaluation metric, BLEU (Papineni et al., 2002), which grades an output candidate according to n-gram matches to one or more reference outputs. To evaluate whether BLEU is an appropriate automatic evaluation measure for response generation, we attempted to measure its agreement with the human judgments. We calculate BLEU using a single reference derived from our parallel corpus. We show the smoothed BLEU 1-4 scores for each system on each dataset evaluated in Table 4. Although these scores are extremely low, the overall BLEU scores agree with overall annotator judgments in all cases except when comparing MT-CHAT and IR-RESPONSE. It would seem that BLEU has some agreement with human judgments on this task, but perhaps not enough to be immediately useful.

7 Conclusions

We have presented several data-driven approaches, based on information retrial and phrase-based SMT, to the task of automatically responding to open-

[6] See inter annotator agreement in table 4.

[7] Also, as one can see from the example exchanges in Table 3, fluency errors are rampant across all systems, including the gold-standard human responses.

Status	MT-CHAT	HUMAN
i want to go to a bonfire right about now.	that does sound fun bonfire i wan na go !	did my tweet bring that upon you ? hehe
I'm the invisible man.	man , you could never be invisible	i can still see your face surrounded in blue !
put ya number in my phone ..and ima call you up !	call me when u do that in your phone call :) for what 's up ?	o well hw kum u dnt eva call or txt me :) ? ! ? ! lolz jpn hw r ya tho :) ? ! ? !
Gray Maynard with hair? I did not approve this.	maynard with grey hair !	i take it back, the hair isn't working for maynard.

Table 5: Examples where MT-CHAT output was preferred over HUMAN response by Turker annotators

domain linguistic stimuli.

Our experiments show that SMT techniques are better-suited than IR approaches on the task of response generation. Our system, MT-CHAT, produced responses which were preferred by human annotators over actual human responses 15% of the time. Although this is still far from human-level performance, we believe there is much room for improvement: from designing appropriate word-alignment and decoding algorithms that account for the selective nature of response in dialogue, to simply adding more training data.

We described the many challenges posed by adapting phrase-based SMT to dialogue, and presented initial solutions to several, including direct phrasal alignment, and phrase-table scores discouraging responses that are lexically similar to the status. Finally, we have provided results from an initial experiment to evaluate the BLEU metric when applied to response generation, showing that though the metric as is does not work well, there is sufficient correlation to suggest that a similar, dialogue-focused approach may be feasible.

By generating responses to Tweets out of context, we have demonstrated that the models underlying phrase-based SMT are capable of guiding the construction of appropriate responses. In the future, we are excited about the role these models could potentially play in guiding response construction for conversationally-aware chat input schemes, as well as goal-directed dialogue systems.

Acknowledgments

We would like to thank Oren Etzioni, Michael Gamon, Jerry Hobbs, Dirk Hovy, Yun-Cheng Ju, Kristina Toutanova, Saif Mohammad, Patrick Pantel, and Luke Zettlemoyer, in addition to the anonymous reviewers for helpful discussions and comments on a previous draft. The first author is suppored by a National Defense Science and Engineering Graduate (NDSEG) Fellowship 32 CFR 168a.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Comput. Linguist.*, 34:555–596, December.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: an entity-based approach. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05.

E. M. Bennett, R. Alpert, and A. C. Goldstein. 1954. Communications through limited-response questioning. *Public Opinion Quarterly*, 18(3):303–308.

Michael Bloodgood and Chris Callison-Burch. 2010. Using mechanical turk to build machine translation evaluation sets. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 208–211, Morristown, NJ, USA. Association for Computational Linguistics.

Chris Brockett. 2006. Aligning the rte 2006 corpus. In *Microsoft Research Techincal report MSR-TR-2007-77*.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Comput. Linguist.*, 16:79–85, June.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 workshop on statistical machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09.

Nathanael Chambers and James Allen. 2004. Stochastic language generation in a dialogue system: Toward a domain independent generator. In Michael Strube and Candy Sidner, editors, *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 9–18, Cambridge, Massachusetts, USA, April 30 - May 1. Association for Computational Linguistics.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words! Linguistic style accommodation in social media. In *Proceedings of WWW*.

Hal Daumé III and Daniel Marcu. 2009. Induction of word and phrase alignments for automatic document summarization. *CoRR*, abs/0907.0804.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Morristown, NJ, USA. Association for Computational Linguistics.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 16–23, Morristown, NJ, USA. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of ACL-08: HLT*, June.

Michel Galley, Eric Fosler-Lussier, and Alexandros Potamianos. 2001. Hybrid natural language generation for spoken dialogue systems. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH–01)*, pages 1735–1738, Aalborg, Denmark, September.

Jon Hasselgren, Erik Montnemery, Pierre Nugues, and Markus Svensson. 2003. Hms: a predictive text entry method using bigrams. In *Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods*, TextEntry '03.

Jerry R. Hobbs. 1985. On the coherence and structure of discourse.

Charles Lee Isbell, Jr., Michael J. Kearns, Dave Kormann, Satinder P. Singh, and Peter Stone. 2000. Cobot in lambdamoo: A social statistics agent. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 36–41. AAAI Press.

Sina Jafarpour and Christopher J. C. Burges. 2010. Filter, rank, and transfer the knowledge: Learning to chat.

Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 967–975, Prague, Czech Republic, June. Association for Computational Linguistics.

Sangkeun Jung, Cheongjae Lee, Kyungduk Kim, Minwoo Jeong, and Gary Geunbae Lee. 2009. Data-driven user simulation for automated evaluation of spoken dialog systems. *Comput. Speech Lang.*, 23:479–509, October.

Kevin Knight and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*. The Association for Computer Linguistics.

J R Landis and G G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*.

Brian Langner, Stephan Vogel, and Alan W. Black. 2010. Evaluating a dialog language generation system: comparing the mountain system to other nlg approaches. In *INTERSPEECH*.

Anton Leuski and David R. Traum. 2010. Practical language processing for virtual humans. In *Twenty-Second Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-10)*.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 802–811, Morristown, NJ, USA. Association for Computational Linguistics.

Robert C. Moore. 2004. On log-likelihood-ratios and the significance of rare events. In *EMNLP*.

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

F. J. Och. 2003. Minimum error rate training for statistical machine translation. In *ACL*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149.

Owen Rambow, Srinivas Bangalore, and Marilyn Walker. 2001. Natural language generation in dialog systems. In *Proceedings of the first international conference on Human language technology research*, HLT '01, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*.

Sujith Ravi, Andrei Broder, Evgeniy Gabrilovich, Vanja Josifovski, Sandeep Pandey, and Bo Pang. 2010. Automatic generation of bid phrases for online advertising. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 464–471, Prague, Czech Republic, June. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 172–180, Morristown, NJ, USA. Association for Computational Linguistics.

Samira Shaikh, Tomek Strzalkowski, Sarah Taylor, and Nick Webb. 2010. Vca: an experiment with a multiparty virtual chat agent. In *Proceedings of the 2010 Workshop on Companionable Dialogue Systems*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06.

Xu Sun, Jianfeng Gao, Daniel Micol, and Chris Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 266–274, Morristown, NJ, USA. Association for Computational Linguistics.

Reid Swanson and Andrew S. Gordon. 2008. Say anything: A massively collaborative open domain story writing companion. In *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling: Interactive Storytelling*, ICIDS '08, pages 32–40, Berlin, Heidelberg. Springer-Verlag.

Lidan Wang and Douglas W. Oard. 2009. Context-based message expansion for disentanglement of interleaved text conversations. In *HLT-NAACL*.

Joseph Weizenbaum. 1966. Eliza: a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9:36–45, January.

Yorick Wilks. 2006. Artificial companions as a new kind of interface to the future internet. In *OII Research Report No. 13*.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*.

Predicting a Scientific Community's Response to an Article

Dani Yogatama Michael Heilman Brendan O'Connor Chris Dyer
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{dyogatama,mheilman,brenocon,cdyer}@cs.cmu.edu

Bryan R. Routledge
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213, USA
routledge@cmu.edu

Noah A. Smith
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nasmith@cs.cmu.edu

Abstract

We consider the problem of predicting measurable responses to scientific articles based primarily on their text content. Specifically, we consider papers in two fields (economics and computational linguistics) and make predictions about downloads and within-community citations. Our approach is based on generalized linear models, allowing interpretability; a novel extension that captures first-order temporal effects is also presented. We demonstrate that text features significantly improve accuracy of predictions over metadata features like authors, topical categories, and publication venues.

1 Introduction

Written communication is an essential component of the complex social phenomenon of science. As such, natural language processing is well-positioned to provide tools for understanding the scientific process, by analyzing the textual artifacts (papers, proceedings, etc.) that it produces. This paper is about modeling collections of scientific documents to understand how their *textual content* relates to how a scientific community responds to them. While past work has often focused on citation structure (Borner et al., 2003; Qazvinian and Radev, 2008), our emphasis is on the text content, following Ramage et al. (2010) and Gerrish and Blei (2010).

Instead of task-independent exploratory data analysis (e.g., topic modeling) or multi-document summarization, we consider supervised models of the collective *response* of a scientific community to a published article. There are many measures of impact of a scientific paper; ours come from direct measurements of the number of downloads (from an established website where prominent economists post papers before formal publication) and citations (within a fixed scientific community). We adopt a discriminative approach based on generalized linear models that can make use of any text or metadata features, and show that simple lexical features offer substantial power in modeling out-of-sample response and in *forecasting* response for future articles. Realistic forecasting evaluations require methodological care beyond the usual best practices of train/test separation, and we elucidate these issues.

In addition, we introduce a new regularization technique that leverages the intuition that the relationship between observable features and response should evolve smoothly over time. This regularizer allows the learner to rely more strongly on more recent evidence, while taking into account a long history of training data. Our time series-inspired regularizer is computationally efficient in learning and is a significant advance over earlier text-driven forecasting models that ignore the time variable altogether (Kogan et al., 2009; Joshi et al., 2010).

We evaluate our approaches in two novel experimental settings: predicting downloads of economics articles and predicting citation of papers at ACL conferences. Our approaches substantially outper-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 594–604,
Edinburgh, Scotland, UK, July 27–31, 2011.

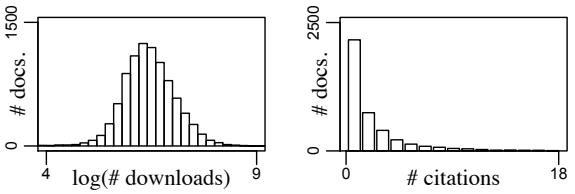


Figure 1: Left: the distribution of log download counts for papers in the NBER dataset one year after posting. Right: the distribution of within-dataset citations of ACL papers within three years of publication (outliers excluded for readability).

form text-ignorant baselines on ground-truth predictions. Our time series models permit flexibility in features and offer a novel and perhaps more interpretable view of the data than summary statistics.

2 Data

We make use of two collections of scientific literature, one from the economics domain, and the other from computational linguistics and natural language processing. Statistics are summarized in Table 1.

2.1 NBER

Our first dataset consists of research papers in economics from the National Bureau of Economic Research (NBER) from 1999 to 2009 (`http://www.nber.org`). Approximately 1,000 research economists are affiliated with the NBER. New NBER working papers are posted to the website weekly. The papers are not yet peer-reviewed, but given the prominence of many economists affiliated with the NBER, many of the papers are widely read. Text from the abstracts of the papers and related metadata are publicly available. Full text is available to subscribers (universities typically have access).

The NBER provided us with download statistics for these papers. For each paper, we computed the total number of downloads in the first year after each paper's posting.[1] The download counts are log-normally distributed, as shown in Figure 1, and so our regression models (§3) minimize squared errors in the log space. Our download logs begin in 1999. We use the 8,814 papers from 1999–2009 period (there are 16,334 papers in the full dataset dating back to 1985). We only use text from the abstracts, since we were able to obtain full texts for just a portion of the papers, and since the OCR of the full texts we do have is very noisy.

Dataset	# Docs.	Avg. # Words	Response
NBER	8,814	155	# downloads in first year (mean 761)
ACL	4,026	3,966	at least 1 citation in first 3 years? (54% no)

Table 1: Descriptive statistics about the datasets.

2.2 ACL

Our second dataset consists of research papers from the Association for Computational Linguistics (ACL) from 1980 to 2006 (Radev et al., 2009a; Radev et al., 2009b). We have the full texts for papers (OCR output) as well as structured citation data. There are 15,689 papers in the whole dataset. For the citation prediction task, we include conference papers from ACL, EACL, HLT, and NAACL.[2] We remove journal papers, since they are characteristically different from conference papers, as well as workshop papers. We do include short papers, interactive demo session papers, and student research papers that are included in the companion volumes for these conferences (such papers are cited less than full papers, but many are still cited). The resulting dataset contains 4,026 papers. The number of papers in each year varies because not all conferences are annual.

We look at citations in the three-year window following publication, excluding self-citations and only considering citations from papers within these conferences. Figure 1 shows a histogram; note that many papers (54%) are not cited at all, and the distribution of citations per paper is neither normal nor log-normal. We organize the papers into two classes: those with zero citations and those with non-zero citations in the three-year window.

[1] For the vast majority of papers, most of the downloads occur soon after the paper's posting. We explored different measures with different download windows (two years, for example) with broadly similar results. We leave a more detailed analysis of the time series patterns of downloads to future work.

[2] EMNLP is a relatively recent conference, and, in this collection, complete data for its papers postdate the end of the last training period, so we chose to exclude it from our dataset.

3 Model

Our forecasting approach is based on generalized linear models for regression and classification. The models are trained with an ℓ_2-penalty, often called a "ridge" model (Hoerl and Kennard, 1970).[3] For the NBER data, where (log) number of downloads is nearly a continuous measure, we use linear regression. For the ACL data, where response is the binary cited-or-not variable we use logistic regression, often referred to as a "maximum entropy" model (Berger et al., 1996) or a log-linear model. We briefly review the class of models. Then, we describe a time series model appropriate for time series data.

3.1 Generalized Linear Models

Consider a model that predicts a response y given a vector input $\boldsymbol{x} = \langle x_1, \ldots, x_d \rangle \in \mathbb{R}^d$. Our models are linear functions of $\boldsymbol{x}$ and parameterized by the vector $\boldsymbol{\beta}$. Given a corpus of M document features, $\boldsymbol{X}$, and responses Y, we estimate:

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin}_{\boldsymbol{\beta}} R(\boldsymbol{\beta}) + \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{X}, Y) \qquad (1)$$

where $\mathcal{L}$ is a model-dependent loss function and R is a regularization penalty to encourage models with small weight vectors. We describe models and loss functions first and then turn to regularization.

For the NBER data, the (log) number of downloads is continuous, and so we use least-squares linear regression model. The loss function is the sum of the squared errors for the M documents in our training data: $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{X}, Y) = \sum_{i=1}^{M} (y_i - \hat{y}_i)^2$, where the prediction rule for new documents is: $\hat{y} = \sum_{j=0}^{d} \beta_j x_j$. Probabilistically, this equates to an assumption that $\boldsymbol{\beta}^\top \boldsymbol{x}$ is the mean of a normal (i.e., Gaussian) distribution from which random variable y is drawn.

For the ACL data, we predict y from a discrete set C (specifically, the binary set of zero citations or more than zero citations), and we use logistic regression. This model assumes that for the ith training input $\boldsymbol{x}_i$, the output y_i is drawn according to:

$$p(y_i \mid \boldsymbol{x}_i) = \left(\exp \boldsymbol{\beta}_c^\top \boldsymbol{x}_i\right) / \left(\textstyle\sum_{c' \in C} \exp \boldsymbol{\beta}_{c'}^\top \boldsymbol{x}_i\right)$$

where there is a feature vector $\boldsymbol{\beta}_c$ for each class $c \in C$. Under this interpretation, parameter estimation is maximum *a posteriori* inference for $\boldsymbol{\beta}$, and $R(\boldsymbol{\beta})$ is a log-prior for the weights. The loss function is the negative log likelihood for the M documents: $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{X}, Y) = -\sum_{i=1}^{M} \log p(y_i \mid \boldsymbol{x}_i)$. The prediction rule for a new document is: $\hat{y} = \operatorname{argmax}_{c \in C} \sum_{j=0}^{d} \beta_{c,j} x_j$. Generalized linear models and penalized regression are well-studied with an extensive literature (Mccullagh and Nelder, 1989; Hastie et al., 2009). We leave other types of models, such as Poisson (Cameron and Trivedi, 1998) or ordinal (McCullagh, 1980) regression models, to future work.

3.2 Ridge Regression

With large numbers of features, regularization is crucial to avoid overfitting. In ridge regression (Hoerl and Kennard, 1970), a standard method to which we compare the time series regularization discussed in §3.3, the penalty $R(\boldsymbol{\beta})$ is proportional to the ℓ_2-norm of $\boldsymbol{\beta}$:

$$R(\beta) = \lambda \|\boldsymbol{\beta}\|_2 = \lambda \textstyle\sum_j \beta_j^2$$

where λ is a regularization hyperparameter that is tuned on development data or by cross-validation.[4] This penalty pushes many β_j close (but not completely) to zero. In practice, we multiply the penalty by the number of examples M to facilitate tuning of λ.

The ridge linear regression model can be interpreted probabilistically as each coefficient β_j is drawn i.i.d. from a normal distribution with mean 0 and variance $2\lambda^{-1}$.

3.3 Time Series Regularization

A simple way to capture temporal variation is to conjoin traditional features with a time variable. Here, we divide the dataset into T time steps (years). In the new representation, the feature space expands from $\mathbb{R}^d$ to $\mathbb{R}^{T \times d}$. For a document published at year t, the elements of $\boldsymbol{x}$ are non-zero only for those features that correspond to year-t; that is $x_{t',j} = 0$ for all $t' \neq t$.

[3] Preliminary experiments found no consistent benefit from ℓ_1 ("lasso") models, though we note that ℓ_1-regularization leads to sparse, compact models that may be more interpretable.

[4] The linear regression has a bias β_0 that is always active. The logistic regression also has an unpenalized bias $\beta_{c,0}$ for each class c. This weight is not regularized.

Estimating this model with the new features using the ℓ_2-penalty would be effectively estimating separate models for each year under the assumption that each $\beta_{t,j}$ is independent; even for features that differed only temporally (e.g., $\beta_{t,j}$ and $\beta_{t+1,j}$).

In this work, we apply time series regularization to GLMs, enabling models that have coefficients that change over time but prefer gradual changes across time steps. Boyd and Vandenberghe (2004, §6.3) describe a general version of this sort of regularizer. To our knowledge, such regularizers have not previously been applied to temporal modeling of text.

The time series regularization penalty becomes:

$$R(\boldsymbol{\beta}) = \lambda \sum_{t=1}^{T} \sum_{j=1}^{d} \beta_{t,j}^2 + \lambda\alpha \sum_{t=2}^{T} \sum_{j=0}^{d} (\beta_{t,j} - \beta_{t-1,j})^2$$

It includes a standard ℓ_2-penalty on the coefficients, and a penalty for differences between coefficients for adjacent time steps to induce smooth changes.[5] Similar to the previous model, in practice, we multiply the regularization constant λ by $\frac{M}{T}$ to facilitate tuning of λ for datasets with different numbers of examples M and numbers of time steps T. The new parameter, α, controls the smoothness of the estimated coefficients. Setting α to zero imposes no penalty for time-variation in the coefficients and results in independent ridge regressions at each time step. Also, when the number of examples is constant across time steps, setting a large α parameter ($\alpha \to \infty$) results in a single ridge regression over all years since it imposes $\beta_{t,j} = \beta_{t+1,j}$ for all $t \in T$.

The partial derivative is:

$$\begin{aligned} \partial R / \partial \beta_{t,j} = & \; 2\lambda\beta_{t,j} \\ & + \; \mathbf{1}\{t > 1\} 2\lambda\alpha(\beta_{t,j} - \beta_{t-1,j}) \\ & + \; \mathbf{1}\{t < T\} 2\lambda\alpha(\beta_{t,j} - \beta_{t+1,j}) \end{aligned}$$

This time series regularization can be applied more generally, not just to linear and logistic regression.

With either ridge regularization or this time series regularization scheme, Eq. 1 is an unconstrained convex optimization problem for the linear models we describe here. There exist a number of optimization procedures for it; we use the L-BFGS quasi-Newton algorithm (Liu and Nocedal, 1989).

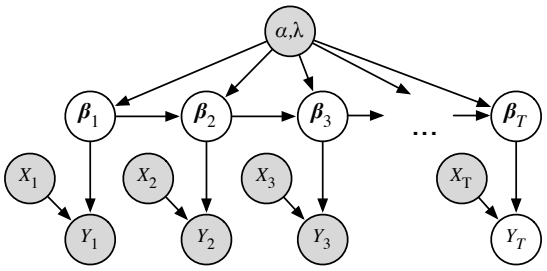


Figure 2: Time series regression as a graphical model; the variables $\boldsymbol{X}_t$ and Y_t are the sets of feature vectors and response variables from documents dated t.

Probabilistic Interpretation

We can interpret the time series regularization probabilistically as follows. Let the coefficient for the jth feature over time be $\boldsymbol{\beta}_j = \langle \beta_{1,j}, \beta_{2,j}, ..., \beta_{T,j} \rangle$. $\boldsymbol{\beta}_j$ are draws from a multivariate normal distribution with a tridiagonal precision matrix $\Sigma^{-1} = \Lambda \in \mathbb{R}^{T \times T}$:

$$\Lambda = \lambda \begin{bmatrix} 1+\alpha & -\alpha & 0 & 0 & \dots \\ -\alpha & 1+2\alpha & -\alpha & 0 & \dots \\ 0 & -\alpha & 1+2\alpha & -\alpha & \dots \\ 0 & 0 & -\alpha & 1+2\alpha & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

The form of $R(\boldsymbol{\beta})$ follows from noting:

$$-2 \log p(\boldsymbol{\beta}_j; \alpha, \lambda) = \boldsymbol{\beta}_j^\top \Lambda \boldsymbol{\beta}_j + \mathit{constant}$$

The squared difference between adjacent time steps comes from the off-diagonal entries in the precision matrix.[6] Figure 2 shows a graphical representation of the time series regularization in our model. Its Markov chain structure corresponds to the off-diagonals.

There is a rich literature on time series analysis (Box et al., 2008; Hamilton, 1994). The prior distribution over the sequence $\langle \beta_{1,j}, \dots, \beta_{T,j} \rangle$ that our regularizer posits is closely linked to a first-order autoregressive process, AR(1).

[5] Our implementation of the time series regularizer does not penalize the magnitude of the weight for the bias feature (as in ridge regression). It does, however, penalize the difference in the bias weight between time steps (as with other features).

[6] Consistent with the previous section, we assume that parameters for different features, $\boldsymbol{\beta}_j$ and $\boldsymbol{\beta}_k$, are independent.

	NBER	ACL
Response	log(#downloads+1)	$\mathbf{1}\{\#\text{citations} > 0\}$
GLM type	normal / squared-loss	logistic / log-loss
Metric 1	mean absolute error	accuracy
Metric 2	Kendall's τ	Kendall's τ

Table 2: Summary of the setup for the NBER download and ACL citation prediction experiments.

4 Features

NBER metadata features

- Authors' last names. We treat each name as a binary feature. If a paper has multiple authors, all authors are used and they have equal weights regardless of their ordering.
- NBER program(s).[7] There are 19 major research programs at the NBER (e.g., Monetary Economics, Health Economics, etc.).

ACL metadata features

- Authors' last names as binary features.
- Conference venues. We use first letter of the ACL anthology paper ID, which correlates with its conference venue (e.g., *P* for the ACL main conference, *H* for the HLT conference, etc.).[8]

Text features

- Binary indicator features for the presence of each unigram, bigram, and trigram. For the NBER data, we have separate features for titles and abstracts. For the ACL data, we have separate features for titles and full texts. We pruned text features by document frequency (details in §5).
- Log transformed word counts. We include features for the numbers of words in the title and the abstract (NBER) or the full text (ACL).

[7] Almost all NBER papers are tagged with one or more programs (we assign untagged papers a "null" tag). The complete list of NBER programs can be found at `http://www.nber.org/programs`

[8] Papers in the ACL dataset have a tag which shows which workshop, conference, or journal they appeared in. However, sometimes a conference is jointly held with another conference, such that meta information in the dataset is different even though the conference is the same. For this reason, we simply use the first letter of the paper ID.

5 Experiments

For each of the datasets in §2, we test our models for two tasks: **forecasting** about future papers (i.e., making predictions about papers that appeared after a training dataset) and **modeling** held-out papers from the past (i.e., making predictions within the same time period as the training dataset, on held-out examples).

For the NBER dataset, the task is to predict the number of downloads a paper will receive in its first year after publication. For the ACL dataset, the task is to predict whether a paper will be cited at all (by another ACL paper in our dataset) within the first three years after its publication. To our knowledge, clean, reliable citation counts are not available for the NBER dataset; nor are download statistics available for the ACL dataset. Table 2 summarizes the variables of interest, model types, and evaluation metrics for the tasks.

5.1 Extrapolation

The lag between a paper's publication and when its outcome (download or citation count) can be observed poses a unique methodological challenge. Consider predicting the number of downloads over g future time steps. If t is the time of forecasting, we can observe the texts of all articles published before t. However, any article published in the interval $[t-g, t]$ is too recent for the outcome measurement of y to be taken. We refer to the interval $[t-g, t]$ as the "forecast gap". Since recent articles are sometimes the most relevant predictions at t, we do not want to ignore them. Consider a paper at time step $t', t-g < t' < t$. To extrapolate its number of downloads, we consider the observed number in $[t', t]$, and then estimate the ratio r of downloads that occur in the first $t-t'$ time steps, against the first g time steps, using the fully observed portion of the training data. We then scale the observed downloads during $[t', t]$ by r^{-1} to extrapolate. The same method is used to extrapolate citation counts.

In preliminary experiments, we observed that extrapolating responses for papers in the forecast gap led to better performance in general. For example, for the ridge regressions trained on all past years with the full feature set, the error dropped from 262 to 259 when using extrapolation compared to with-

out extrapolation. Also, the extrapolated download counts were quite close to the true values (which we have but do not use because of the forecast gap): for example, the mean absolute error of the extrapolated responses was 99 when extrapolated based on the median of the fully observed portion of the training data (measured monthly).

5.2 Forecasting NBER Downloads

In our first set of experiments, we predict the number of downloads of an NBER paper within one year of its publication.

We compare four approaches for predicting downloads. The first is a baseline that simply uses the median of the log of the training and development data as the prediction. The second and third use GLMs with ridge regression-style regularization (§3.2), trained on all past years ("all years") and on the single most recent past year ("one year"), respectively. The last model ("time series") is a GLM with time series regularization (§3.3).

We divided papers by year. Figure 3 illustrates the experimental setup. We held out a random 20% of papers for each year from 1999–2007 as a test set for the task of modeling the past. To define the feature set and tune hyperparameters, we used the remaining 80% of papers from 1999–2005 as our training data and the remaining papers in 2006 as our development data. After pruning,[9] we have 37,251 total features, of which 2,549 are metadata features. When tuning hyperparameters, we *simulated* the existence of a forecast gap by using extrapolated responses for papers in the last year of the training data instead of their true responses. We considered $\lambda \in 5^{\{2,1,\ldots,-5,-6\}}$, and $\alpha \in 5^{\{3,2,\ldots,-1,-2\}}$ and selected those that led to the best performance on the development set.

We then used the selected feature set and hyperparameters to test the forecasting and modeling capabilities of each model. For forecasting, we predicted numbers of downloads of papers in 2008 and 2009. We used the baseline median, ridge regression, and time series regularization models trained on papers in 1999–2007 and 1999–2008, respectively. We treated the last year of the training data (2007 and 2008, respectively) as a forecast gap, since we would not have observed complete responses of papers in these years when forecasting. For the "one year" models, we trained ridge regressions only on the most recent past year, using papers in 2007 and 2008, respectively, as training data.[10] To test the additive benefit of text features, we trained models with just metadata features (NBER programs and authors, denoted "Meta") and with both metadata

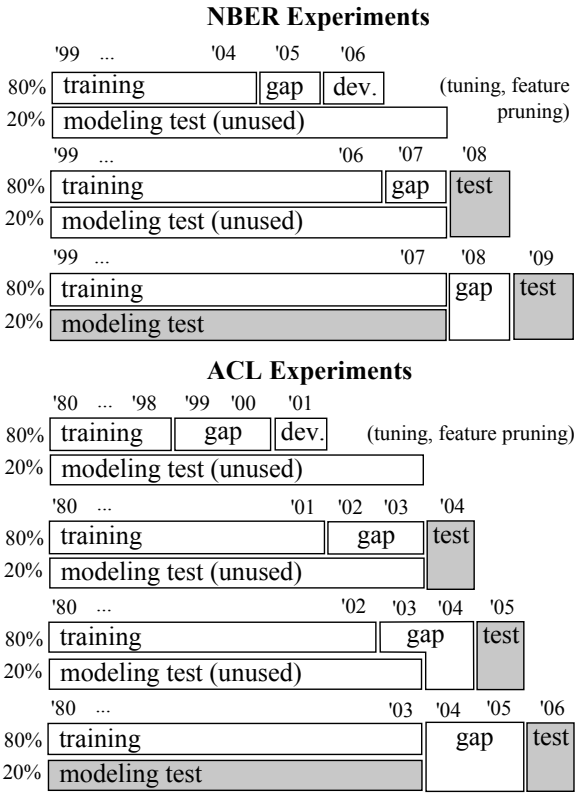


Figure 3: An illustration of how the datasets were segmented for the experiments. Portions of data for which we report results are shaded. Time spans are not to scale.

[9] For NBER, text features appearing in less than 0.1% or more than 99.9% of the training documents were removed. For ACL, the thresholds were 2% and 98%.

[10] Papers from the most recent past year in a training set have incomplete responses, so the models were trained on extrapolated responses for that year. For the NBER development set from 2005, a ridge regression on just 2004 papers (for which extrapolation is needed) outperformed a regression on just 2003 (for which extrapolation is not needed), 278 to 367 mean absolute error. For the ACL development set from 2001, a regression on just 2000 (for which extrapolation is needed) led to slightly lower performance (59% versus 61%) than a regression on just 1998 (for which extrapolation is not needed), probably due to the relatively small number of conferences and papers in 2000. For consistency with the other models and with the NBER experiments, we evaluated regressions on the most recent (extrapolated) year in our ACL experiments.

Features	Model	Modeling 1999–07	Forecasting 2008	Forecasting 2009
–	median	333	371	397
Meta	one year	279	354	375
Meta	all years	303	334	378
Meta	time series	279	353	375
Full	one year	271	346	351
Full	all years	265	†**300**	339
Full	time series	*†**245**	*321	***332**

Table 3: Mean absolute errors for the NBER download predictions. "$*$" indicates statistical significance between time series models using metadata features and the full feature set. "†" indicates statistical significance between the time series and ridge regression models using the full feature set (Wilcoxon signed-rank test, $p < 0.01$).

Feat.	Model	Modeling 1980–03	Forecasting 2004	Forecasting 2005	Forecasting 2006
–	majority	55	56	60	50
Meta	one year	61	56	54	62
Meta	all years	65	58	53	60
Meta	time series	66	56	53	56
Full	one year	69	**70**	64	67
Full	all years	67	69	**70**	70
Full	time series	**70**	*69	***70**	***72**

Table 4: Classification accuracy (%) for predicting whether ACL papers will be cited within three years. "$*$" indicates statistical significance between time series models using metadata features and the full feature set (binomial sign test, $p < 0.01$). With the full feature set, differences between the time series and ridge (all years) models are not statistically significant at the 0.01 level, but for the modeling task p is estimated at 0.026, and for the 2006 forecasting task, p is estimated at 0.050.

and text features (denoted "Full").

To evaluate the modeling capabilities, we trained the ridge regression and time series regularization models on papers from 1999–2008 and predicted the numbers of downloads of held-out papers in 1999–2007. For comparison, we also trained ridge regression models on each individual year ("one year") and predicted the numbers of downloads of the held-out papers in the corresponding year.

Table 3 shows mean absolute errors for each method on both forecasting test splits, and mean absolute errors averaged across papers over nine modeling test splits. For interpretability, we report predictions in terms of download counts, though the models were trained with log counts (§2.1). The results show that even a simple n-gram representation of text contains a valuable, learnable signal that is predictive of future downloads. While the time series model did not significantly outperform ridge regression at predicting future downloads, it did result in significantly better performance for *modeling* papers in the past.

5.3 Forecasting ACL Citations

We now turn to the problem of predicting citation levels. Recall that here we aim to predict whether an ACL paper will be cited within our dataset within three years. Our experimental setup (Figure 3) is similar to the setup for the NBER dataset, except that we use logistic regression to model the discrete cited-or-not response variable. We also make the simplifying assumption that all citations occur at the end of each year. Therefore, the forecast gap is only two years (we have observed complete citations in the test year).

After feature pruning, there were 30,760 total features, of which 1,694 are metadata features. We considered $\lambda \in 5^{\{2,1,\cdots,-8,-9\}}$ ("Full") and $\lambda \in 5^{\{2,1,\cdots,-11,-12\}}$ ("Meta"); and $\alpha \in 5^{\{6,5,\cdots,0,-1\}}$ (both "Full" and "Meta"), selecting the best values using the development data.

Again, we compare four methods: a baseline of always predicting the most frequent class in the training data, "all years" and "one year" logistic regression models, and a logistic regression with the time series regularizer.

For the forecasting task, we used papers in 2004, 2005, and 2006 as test sets. As the training sets for the "all years" and time series models, we used papers from 1980 up to the last year before each test set, with the last two years extrapolated. As the training sets for the "one year" models, we used papers from the year immediately before the test set, with extrapolated responses.

To evaluate modeling capabilities, we predicted citation levels of held-out papers in 1980–2003. We used the "all years" and time series models trained on 1980–2005. We trained "one year" models separately for each year and predicted downloads for the held-out papers in that year.

Table 4 shows classification accuracy for each model on the test data for both the forecasting and modeling tasks. It is again clear that adding text sig-

nificantly improved the performance of the model. Also, the time series regression model shows a small, though not statistically significant, gain for modeling whether past papers will be cited—as well as similarly small gains on two of the three forecasting test years.

5.4 Ranking

We can also use the models for ranking to help decide which papers are expected to have the greatest impact. With rankings, we can use the same metric both for download and citation predictions. For the NBER data, we ranked test-set papers based on the predicted numbers of downloads and computed the correlation to the actual numbers of downloads. For the ACL data, we ranked papers based on the *probability* of being cited (within the next three years) and computed the correlation to the actual numbers of citations.[11]

To measure ranking models' ranking quality, we used Kendall's τ, a nonparametric statistic that measures the similarity of two different orderings over the same set of items. Here, the items are scientific papers and the two metrics are the gold standard numbers of downloads (or citations) and model predictions for the numbers of downloads, or citation probabilities. If q is the chance that a randomly drawn pair of items will be ranked in the same way by the two metrics, then $\tau = 2(q - 0.5)$.

Table 5 shows Kendall's τ for each model for the forecasting tasks (i.e., prediction of future citations or downloads) in both datasets. As in the previous experiments, we see small benefits for the time series regression model on most held-out data splits—and larger benefits for including text features along with metadata features.

6 Analysis

An advantage of the time series regularized regression model is its interpretability. Inspecting feature coefficients in the model allows us to identify trends and changes of interests over time within a scientific community.

[11] Here, we use models of responses to individual papers for ranking (i.e., in a pointwise ranking scheme). Time series regularization could also be applied to ranking models that model pairwise preferences to optimize metrics like Kendall's τ directly, as discussed by Joachims (2002).

Feat.	Model	NBER		ACL		
		'08	'09	'04	'05	'06
Meta	one year	.29	.22	.17	.08	.16
Meta	all years	.31	.22	.15	.12	.21
Meta	time series	.29	.22	.14	.10	.17
Full	one year	.35	.31	.44	.39	.33
Full	all years	**.43**	.37	.42	.43	.40
Full	time series	**.43**	**.38**	**.47**	**.44**	**.43**

Table 5: Kendall's τ rank correlation for future prediction models on both datasets.

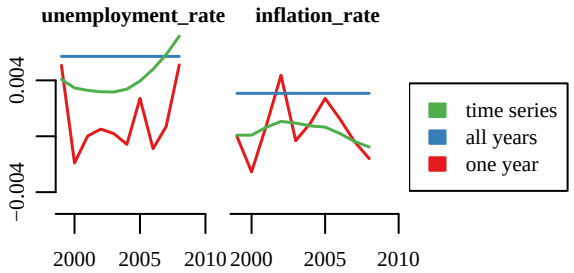


Figure 4: Coefficients for two NBER bigram features.

First, we illustrate the difference between the time series and the other models in Figure 4, for NBER models' weights for *unemployment rate* and *inflation rate* appearing in a paper's abstract. The year-to-year weights of "one year" models fluctuate substantially, and the "all years" model is necessarily constant, but the time series regularizer gives a smooth trajectory.

6.1 Trends

Previous work has examined the flow of ideas as trends in word and phrase frequencies, as in the Google Books Ngram Viewer (Michel et al., 2011).[12] Topic models have been used extensively to explore trends in low-dimensional spaces (Blei and Lafferty, 2006; Wang et al., 2008; Wang and McCallum, 2006; Ahmed and Xing, 2010). By contrast, our approach allows us to examine trends in the *impact* of text related to specific observation variables: the coefficient trendline for a feature illustrates its association with measurements of scholarly impact (citation and download frequency).

Text frequencies can be quite different from the discriminative weights our model assigns to features. Figure 5 illustrates the $\beta_{t,j}$ trends in the ACL time series model for some selected terms that oc-

[12] http://ngrams.googlelabs.com

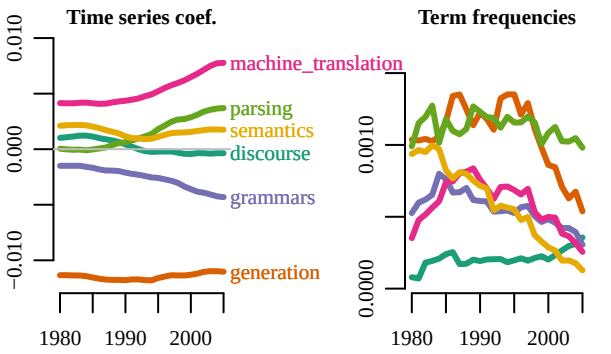
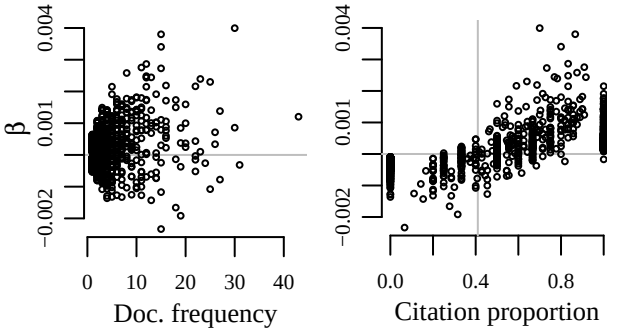


Figure 5: Feature trends: model coefficients vs. term frequencies over time in the ACL corpus. Term freq. is the fraction of tokens (or bigrams for *m.t.*) that year, that are the term, averaged over a centered five-year window.



(a) Doc. freq. vs. coef. (b) Citation prop. vs. coef.

Figure 6: Analysis of author citation coefficients. Every point is one ACL author, and the vertical axis shows the citation coefficient, compared to (a) the number of documents co-authored by the author; and (b) the proportion of an author's papers that are cited within three years. The vertical bar is the *macro*-averaged citation proportion across authors, 41%.

cur frequently in conference session titles. On the right are term frequencies (with smoothing, since year-to-year frequencies are bumpy). Most terms decline over time. On the left, by contrast, are the weights learned by our time series model. They tell a very different story: for example, parsing has shown a definite increase in interest, while interest in grammars (e.g., formalisms) has declined somewhat. These trends have face validity, giving credence to our analysis; they also broadly agree with Hall et al. (2008).

6.2 Authors

The regression method also allows analysis of author influence, since we fit a coefficient for each of the authors in the ACL dataset. Figure 6(a) addresses the following question: do prolific authors get cited more often, even after accounting for the content of their papers?[13] The effect is present but relatively small according to our model: the total number of papers co-authored by an author has a weak correlation to the author's citation prediction coefficient ($\tau = 0.16$).

Next, does the model provide more information than the simple citation probability of an author? Figure 6(b) compares coefficients to an author's papers' probability of being cited. Since we did not prune author features, there are many authors with only a few papers, resulting in unsmoothed probabilities of 0, 0.5, 1, etc. (these correspond to the vertical "bands" in the plot). By contrast, the ℓ_2-penalty of the model naturally assigned coefficients close to zero for such authors if it is justified.

In general, the simple probability agrees with the coefficient, but there are differences. The semantics of the regression imply we are measuring the relative citation probability of an author, *controlling* for text and venue effects. If an author has a high citation prediction coefficient but a low citation probability, that implies the author has better-cited work than would be expected according to the n-grams in his or her papers. We have omitted names of authors from the figure for clarity and confidentiality, but high outlier authors tend to be well-known researchers in the ACL community. Obviously, since the prediction model is not perfect, it is not possible to completely verify this hypothesis, but we feel this analysis is reasonably suggestive.

7 Related Work

Previous work on modeling scientific literature mostly focused on citation graphs (Borner et al., 2003; Qazvinian and Radev, 2008). Some researchers, e.g., Erosheva et al. (2004), have used text content. Most of these are based on topic models: Gerrish and Blei (2010) measure scholarly impact, Hall et al. (2008) study the "history of ideas",

[13] More precisely: if a prolific author and a non-prolific author write a paper, does the prolific author's paper have a higher probability of being cited than the non-prolific author's, all other things being equal?

and Ramage et al. (2010) rank universities based on scholarly output using topic models.

Download rates and citation prediction were two of the main tasks in the KDD Cup 2003 (McGovern et al., 2003; Brank and Leskovec, 2003). Bethard and Jurafsky (2010) considered the problem slightly differently and proposed an information retrieval approach to citation prediction. Our approach is novel in that we formulate the problem as a forecasting task and we seek to predict *future* impact of articles.

Linear regression with text features has been used to predict financial risk (Kogan et al., 2009) and movie revenues (Joshi et al., 2010). While the forecasts in those papers are similar to ours, those authors did not consider a forecast gap or allowing the parameters of the model to vary over time.

Our time series regularization is closely related to the fused lasso (Tibshirani et al., 2005). It penalizes a loss function by the ℓ_1-norm of the coefficients and their differences. The ℓ_1-penalty for differences between coefficients encourages *sparsity* in the differences. We use the ℓ_2-norm to induce *smooth* changes across time steps.

8 Conclusions

We presented a statistical approach to predicting a scientific community's response to an article, based on its textual content. To improve the interpretability of the linear model, we developed a novel time series regularizer that encourages gradual changes across time steps. Our experiments showed that text features significantly improve accuracy of predictions over baseline models, and we found that the feature weights learned with the time series regularizer reflect important trends in the literature.

Acknowledgements

We thank the National Bureau of Economic Research for providing the NBER dataset for this research, Fallaw Sowell for helpful discussions, and three anonymous reviewers for comments on an earlier draft of this paper. This research was supported by the Intelligence Advanced Research Projects Activity under grant number N10PC20222 and TeraGrid resources provided by the Pittsburgh Supercomputing Center under grant number TG-DBS110003.

References

A. Ahmed and E. P. Xing. 2010. Timeline: A dynamic hierarchical Dirichlet process model for recovering birth/death and evolution of topics in text stream. In *Proc. of UAI*.

A. L. Berger, V. J. Della Pietra, and S. A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

S. Bethard and D. Jurafsky. 2010. Who should I cite? Learning literature search models from citation behavior. In *Proc. of CIKM*.

D. Blei and J. Lafferty. 2006. Dynamic topic models. In *Proc. of ICML*.

K. Borner, C. Chen, and K. Boyack. 2003. Visualizing knowledge domains. In B. Cronin, editor, *Annual Review of Information Science and Technology*, volume 37, pages 179–255. Information Today, Inc.

G. Box, G. M. Jenkins, and G. Reinsel. 2008. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics.

S. Boyd and L. Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

J. Brank and J. Leskovec. 2003. The download estimation task on KDD Cup 2003. *SIGKDD Explorations*, 5(2):160–162.

A. Cameron and P. Trivedi. 1998. *Regression Analysis of Count Data*. Cambridge University Press.

E. Erosheva, S. Fienberg, and J. Lafferty. 2004. Mixed membership models of scientific publications. In *Proc. of PNAS*.

S. Gerrish and D. M. Blei. 2010. A language-based approach to measuring scholarly impact. In *Proc. of ICML*.

D. Hall, D. Jurafsky, and C. D. Manning. 2008. Studying the history of ideas using topic models. In *Proc. of EMNLP*.

J. D. Hamilton. 1994. *Time Series Analysis*. Princeton University Press.

T. Hastie, R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

A. E. Hoerl and R. W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proc. of KDD*.

M. Joshi, D. Das, K. Gimpel, and N. A. Smith. 2010. Movie reviews and revenues: An experiment in text regression. In *Proc. of HLT-NAACL*.

S. Kogan, D. Levin, B. R. Routledge, J. S. Sagi, and N. A. Smith. 2009. Predicting risk from financial reports with regression. In *Proc. of HLT-NAACL*.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

P. Mccullagh and A. J. Nelder. 1989. *Generalized Linear Models*. London: Chapman & Hall.

P. McCullagh. 1980. Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42(2):109–142.

A. McGovern, L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville, and D. Jensen. 2003. Exploiting relational structure to understand publication patterns in high-energy physics. *SIGKDD Explorations*, 5(2):165–172.

J. Michel, Y. Shen, A. Aiden, A. Veres, M. Gray, The Google Books Team, J. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. Nowak, and E. Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

V. Qazvinian and D. R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proc. of COLING*.

D. R. Radev, M. T. Joseph, B. Gibson, and P. Muthukrishnan. 2009a. A bibliometric and network analysis of the field of computational linguistics. *Journal of the American Society for Information Science and Technology*.

D. R. Radev, P. Muthukrishnan, and V. Qazvinian. 2009b. The ACL anthology network corpus. In *Proc. of ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*.

D. Ramage, C. D. Manning, and D. A. McFarland. 2010. Which universities lead and lag? Toward university rankings based on scholarly output. In *Proc. of NIPS Workshop on Computational Social Science and the Wisdom of the Crowds*.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. 2005. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society B*, 67(1):91–108.

X. Wang and A. McCallum. 2006. Topics over time: A non-Markov continuous-time model of topical trends. In *Proc. of KDD*.

C. Wang, D. Blei, and D. Heckerman. 2008. Continuous time dynamic topic models. In *Proc. of UAI*.

Non-parametric Bayesian Segmentation of Japanese Noun Phrases

Yugo Murawaki and **Sadao Kurohashi**
Graduate School of Informatics
Kyoto University
{murawaki, kuro}@i.kyoto-u.ac.jp

Abstract

A key factor of high quality word segmentation for Japanese is a high-coverage dictionary, but it is costly to manually build such a lexical resource. Although external lexical resources for human readers are potentially good knowledge sources, they have not been utilized due to differences in segmentation criteria. To supplement a morphological dictionary with these resources, we propose a new task of Japanese noun phrase segmentation. We apply non-parametric Bayesian language models to segment each noun phrase in these resources according to the statistical behavior of its supposed constituents in text. For inference, we propose a novel block sampling procedure named hybrid type-based sampling, which has the ability to directly escape a local optimum that is not too distant from the global optimum. Experiments show that the proposed method efficiently corrects the initial segmentation given by a morphological analyzer.

1 Introduction

Word segmentation is the first step of natural language processing for Japanese, Chinese and Thai because they do not delimit words by white-space. Segmentation for Japanese is a successful field of research, achieving the F-score of nearly 99% (Kudo et al., 2004). This success rests on a high-coverage dictionary. Unknown words, or words not covered by the dictionary, are often misidentified.

Historically, researchers have devoted extensive human resources to build and maintain high-coverage dictionaries (Yokoi, 1995). Since the orthography of Japanese does not specify a standard for segmentation, researchers define their own criteria before constructing lexical resources. For this reason, it is difficult to exploit existing external resources, such as dictionaries and encyclopedias for human readers, where entry words are not segmented according to the criteria. Among them, encyclopedias are especially important in that they contain a lot of terms that a morphological dictionary fails to cover. Most of these terms are noun phrases and consist of more than one word (morpheme). For example, an encyclopedia has an entry "常山城" (*tsuneyama-jou*, "Tsuneyama Castle"). According to our segmentation criteria, it consists of two words "常山" (*tsuneyama*) and "城" (*jou*). However, the morphological analyzer wrongly segments it into "常" (*tsune*) and "山城" (*yamashiro*) because "常山" (*tsuneyama*) is an unknown word.

In this paper, we present the first attempt to utilize encyclopedias for word segmentation. We segment each entry noun phrase into words. To do this, we examine the main text of the entry, on the assumption that if the noun phrase in question consists of more than one word, its constituents appear in the main text either freely or as part of other noun phrases. For "常山城" (*tsuneyama-jou*), its constituent "常山" (*tsune*) appears by itself and as constituents of other nouns phrases such as "常山山頂" (peak of Tsuneyama) and "常山駅" (Tsuneyama Station) while "山城" (*yamashiro*) does not.

To segment each noun phrase, we use non-parametric Bayesian language models (Goldwater et al., 2009; Mochihashi et al., 2009). Our approach

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 605–615,
Edinburgh, Scotland, UK, July 27–31, 2011.

is based on two key factors: the bigram model and type-based block sampling. The bigram model alleviates a problem of the unigram model, that is, a tendency to misidentify a sequence of words in common collocations as a single word. Type-based sampling (Liang et al., 2010) has the ability to directly escape a local optimum, making inference very efficient. However, type-based sampling is not easily applicable to the bigram model owing to sparsity and its dependence on latent assignments.

We propose a hybrid type-based sampling procedure, which combines the Metropolis-Hastings algorithm with Gibbs sampling. We circumvent the sparsity problem by joint sampling of unigram-level type. Also, instead of calculating the probability of every possible state of the jointly sampled random variables, we only compare the current state with a proposed state. This greatly eases the sampling procedure while retaining the efficiency of type-based sampling. Experiments show that the proposed method quickly corrects the initial segmentation given by a morphological analyzer.

2 Related Work

Japanese Morphological Analysis and Lexical Acquisition Word segmentation for Japanese is usually solved as the joint task of segmentation and part-of-speech tagging, which is called morphological analysis (Kurohashi et al., 1994; Asahara and Matsumoto, 2000; Kudo et al., 2004). The standard approach in Japanese morphological analysis is lattice-based path selection instead of character-based IOB tagging. Given a sentence, an analyzer first builds a lattice of words with dictionary look-up and then selects an optimal path using pre-defined parameters. This approach enables fast decoding and achieves accuracy high enough for practical use.

This success, however, depends on a high-coverage dictionary, and unknown words are often misidentified. Although a line of research attempts to identify unknown words on the fly (Uchimoto et al., 2001; Asahara and Matsumoto, 2004), it by no means provides a definitive solution because it suffers from locality of contextual information available for identification (Nakagawa and Matsumoto, 2006). Therefore we like to perform separate lexical acquisition processes in which wider context can be examined.

Our approach in this paper has a complementary relationship with unknown word acquisition from text, which we previously proposed (Murawaki and Kurohashi, 2008). Since, unlike Chinese and Thai, Japanese is rich in morphology, morphological regularity can be used to determine if an unknown word candidate in text is indeed the word to be acquired. In general, this method works pretty well, but one exception is noun phrases. Noun phrases can hardly be distinguished from single nouns because in Japanese, no morphological marker is attached to join nouns to form a noun phrase. We previously resort to a heuristic measure to segment noun phrases. The new statistical method provides a straightforward solution to this problem.

Meanwhile, our language models have their own problem. The assumption that language is a sequence of invariant words fails to capture rich morphology, as our segmentation criteria specify that each verb or adjective consists of an invariant stem and an ending that changes its form according to its grammatical roles. For this reason, we limit our scope to noun phrases in this paper.

Use of Noun Phrases Named entity recognition (NER) is a field where encyclopedic knowledge plays an important role. Kazama and Torisawa (2008) encode information extracted from a gazetteer (e.g. Wikipedia) as features of a CRF-based Japanese NE tagger. They formalize the NER task as the character-based labeling of IOB tags. Noun phrases extracted from a gazetteer are also straightforwardly represented as IOB tags. However, this does not fully solve the knowledge bottleneck problem. They also used the output of a morphological analyzer, which does not utilize encyclopedic knowledge. NER performance may be affected by segmentation errors in morphological analysis involving unknown words.

Chinese word segmentation is often formalized as a character tagging problem (Xue, 2003). In this setting, it is easy to incorporate external resources into the model. Low et al. (2005) introduce an external dictionary as features of a discriminative model. However, they only use words up to 4 characters in length. We conjecture that words in their dictionary are not noun phrases. External resources used by

Peng et al. (2004) are also lists of short words and characters.

Non-parametric Language Models Non-parametric Bayesian statistics offers an elegant solution to the task of unsupervised word segmentation, in which the vocabulary size is not known in advance (Goldwater et al., 2009; Mochihashi et al., 2009). It does not compete with supervised segmentation, however. Unsupervised word segmentation is used elsewhere, for example, with theoretical interest in children's language acquisition (Johnson, 2008; Johnson and Demuth, 2010) and with the application to statistical machine translation, in which segmented text is merely an intermediate representation (Xu et al., 2008; Nguyen et al., 2010). In this paper we demonstrate that non-parametric models can complement supervised segmentation.

3 Japanese Noun Phrase Segmentation

Our goal is to overcome the unknown word problem in morphological analysis by utilizing existing resources such as dictionaries and encyclopedias for human readers. In our settings, we are given a list of entries from external resources. Almost all of them are noun phrases and each entry consists of one or more words.

A naïve implementation would be to use noun phrases as they are. In fact, ipadic[1] regards as single words a large number of long proper nouns like "関西国際空港会社連絡橋" (literally, Kansai International Airport Company Connecting Bridge). However, this approach has various drawbacks. For example, in information retrieval, the query "Kansai International Airport" does not match the "single" word for the bridge. So we apply segmentation.

Each entry is associated with text, which is usually the main text of the entry.[2] We assume the text as the key to segmenting the noun phrase. If the noun phrase in question consists of more than one word, its constituents would appear in the text either freely or as part of other noun phrases.

We obtain the segmentation of an entry noun phrase by considering the segmentation of the whole text. One may instead consider a pipeline approach in which we first extract noun phrases in text and then identify boundaries within these noun phrases. However, noun phrases in text are not trivially identifiable in the case that they contain unknown words as their constituents. For example, the analyzer erroneously segments the word "ちんすこう" (*chiNsukou*) into "ちん" (*chiN*) and "すこう" (*sukou*), and since the latter is misidentified as a verb, the incorrect noun phrase "ちん" (*chiN*) is extracted.

We have a morphological analyzer with a dictionary that covers frequent words. Although it often misidentifies unknown words, the overall accuracy is reasonably high. For this reason, we like to use the segmentation given by the analyzer as the initial state and to make small changes to them to get a desired output. We also use an annotated corpus, which was used to build the analyzer. As the annotated corpus encodes our segmentation criteria, it can be used to force the models to stick with our segmentation criteria.

We concentrate on segmentation in this paper, but we also need to assign a POS tag to each constituent word and to incorporate segmented noun phrases into the dictionary of the morphological analyzer. We leave them for future work.[3]

4 Non-parametric Bayesian Language Models

To correct the initial segmentation given by the analyzer, we use non-parametric Bayesian language models that have been applied to unsupervised word segmentation (Goldwater et al., 2009). Specifically, we adopt unigram and bigram models. We propose a small modification to these models in order to exploit an annotated corpus when it is much larger than raw text.

4.1 Unigram Model

In the unigram model, a word in the corpus w_i is generated as follows:

$$\begin{aligned} G|\alpha_0, P_0 &\sim \mathrm{DP}(\alpha_0, P_0) \\ w_i|G &\sim G \end{aligned}$$

[1] http://sourceforge.jp/projects/ipadic/

[2] We may augment the text with related documents if the main text is not large enough.

[3] Fortunately, the morphological analyzer JUMAN is capable of handling *phrases*, each of which consists of more than one word. All we need to do is POS tagging.

where G is a distribution over a countably infinite set of words, and $\mathrm{DP}(\alpha_0, P_0)$ is a Dirichlet process (Ferguson, 1973) with the concentration parameter α_0 and the base distribution P_0, for which we use a zerogram model described in Section 4.3.

Marginalizing out G, we can interpret the model as a Chinese restaurant process. Suppose that we have observed $i-1$ words $\mathbf{w_{-i}} = w_1, \cdots, w_{i-1}$, the probability of w_i is given by

$$P_1(w_i = w|\mathbf{w_{-i}}) = \frac{n_w^{\mathbf{w_{-i}}} + \alpha_0 P_0}{i - 1 + \alpha_0}, \qquad (1)$$

where $n_w^{\mathbf{w_{-i}}}$ is the number of word label w observed in $\mathbf{w_{-i}}$.

The unigram model is known for its tendency to misidentify a sequence of words in common collocations as a single word (Goldwater et al., 2009). In preliminary experiments, we found that the unigram model often interpreted a noun phrase as a single word, even in the case that its constituents frequently appeared in text.

4.2 Bigram Model

The problem of the unigram model can be alleviated by the bigram model based on a hierarchical Dirichlet process (Goldwater et al., 2009). In the bigram model, word w_i is generated as follows:

$$\begin{aligned} G|\alpha_0, P_0 &\sim \mathrm{DP}(\alpha_0, P_0) \\ H_l|\alpha_1, G &\sim \mathrm{DP}(\alpha_1, G) \\ w_i|w_{i-1} = l, H_l &\sim H_l \end{aligned}$$

Marginalizing out G and H_l, we can again explain the model with the Chinese restaurant process. Unlike the unigram model, however, the bigram model depends on the latent table assignments $\mathbf{z_{-i}}$.

$$P_2(w_i|\mathbf{h_{-i}}) = \frac{n_{(w_{i-1},w_i)}^{\mathbf{h_{-i}}} + \alpha_1 P_1(w_i|\mathbf{h_{-i}})}{n_{(w_{i-1},*)}^{\mathbf{h_{-i}}} + \alpha_1} \qquad (2)$$

$$P_1(w_i|\mathbf{h_{-i}}) = \frac{t_{w_i}^{\mathbf{h_{-i}}} + \alpha_0 P_0(w_i)}{t_*^{\mathbf{h_{-i}}} + \alpha_0} \qquad (3)$$

where $\mathbf{h_{-i}} = (\mathbf{w_{-i}}, \mathbf{z_{-i}})$, $t_{w_i}^{\mathbf{h_{-i}}}$ is the number of tables labeled with w_i and $t_*^{\mathbf{h_{-i}}}$ is the total number of tables. Thanks to exchangeability, we do not need to track the exact seating assignments. Still, we need to maintain a *histogram* for each w that consists of frequencies of table customers (Blunsom et al., 2009).

4.3 Zerogram Model

Following Nagata (1996) and Mochihashi et al. (2009), we model the zerogram distribution P_0 with the word length k and the character sequence $w = c_1, \cdots, c_k$. Specifically, we define P_0 as the combination of a Poisson distribution with mean λ and a bigram distribution over characters.

$$\begin{aligned} P_0(w) &= P(k;\lambda)\frac{P(c_1, \cdots, c_k, k|\Theta)}{P(k|\Theta)} \\ P(k;\lambda) &= e^{-\lambda}\frac{\lambda^k}{k!} \\ P(c_1, \cdots, c_k, k|\Theta) &= \prod_{i=1}^{k+1} P(c_i|c_{i-1}) \end{aligned}$$

Θ is the zerogram model, and c_0 and c_{k+1} are a word boundary marker. $P(k|\Theta)$ can be estimated by randomly generating words from the model. We use different λ for different scripts. The Japanese writing system uses several scripts, and each word can be classified by script such as hiragana, katakana, kanji, the mixture of hiragana and kanji, etc. The optimal value for λ depends on scripts. For example, katakana, which predominantly denotes loan words, is longer on average than hiragana, which is often used for short function words.

We obtain the parameters and counts from an annotated corpus and fix them during noun phrase segmentation. This greatly simplifies inference but may make the model fragile with unknown words. For this reason, we set a hierarchical Pitman-Yor process prior (Teh, 2006; Goldwater et al., 2006) for the bigram probability $P(c_i|c_{i-1})$ with the base distribution of character unigrams. Note that even character bigrams are sparse because thousands of characters are used in Japanese.

4.4 Mixing an Annotated Corpus

An annotated corpus can be used to force the models to stick with our segmentation criteria. A straightforward way to do this is to mix it with raw text while fixing the segmentation during inference (Mochihashi et al., 2009). A word found in the annotated corpus is generally preferred because it has fixed counts obtained from the annotated corpus. We call this method direct mixing.

Direct mixing is problematic when raw text is much smaller than the annotated corpus. With this

situation, the role of raw text associated with the noun phrase in question is marginalized by the annotated corpus.

As a solution to this problem, we propose another mixing method called back-off mixing. In back-off mixing, the annotated corpus is used as part of the base distribution. In the unigram model, P_0 in (1) is replaced by

$$P_0^{\mathrm{BM}} = \lambda_{\mathrm{IP}} P_0 + (1 - \lambda_{\mathrm{IP}}) P_1^{\mathrm{REF}},$$

where λ_{IP} is a parameter for linear interpolation and P_1^{REF} is the unigram probability obtained from the annotated text. The loose coupling makes the models robust to an imbalanced pair of texts. Similarly, the back-off mixing bigram model replaces P_1 in (2) with

$$P_1^{\mathrm{BM}} = \lambda_{\mathrm{IP}} P_1 + (1 - \lambda_{\mathrm{IP}}) P_2^{\mathrm{REF}}.$$

5 Inference

Collapsed Gibbs sampling is widely used to find an optimal segmentation (Goldwater et al., 2009). In this section, we first show that simple collapsed sampling can hardly escape the initial segmentation. To address this problem, we apply a block sampling algorithm named type-based sampling (Liang et al., 2010) to the unigram model. Since type-based sampling is not applicable to the bigram model, we propose a novel sampling procedure for the bigram model, which we call hybrid type-based sampling.

5.1 Collapsed Sampling

In collapsed Gibbs sampling, the sampler repeatedly samples every possible boundary position, conditioned on the current state of the rest of the corpus. It stochastically decides whether the corresponding local area consists of a single word w_1 or two words $w_2 w_3$ ($w_1 = w_2.w_3$). The conditional probabilities can be derived from (1).

Collapsed sampling is known for slow convergence. This property is especially problematic in our settings where the initial segmentation is given by a morphological analyzer. Since the analyzer deterministically segments text using pre-defined parameters, the resultant segmentation is fairly consistent. Segmentation errors involving unknown words also occur in a regular way. Intuitively, we start with a local optimum although it is not too distant from the global optimum. The collapsed Gibbs sampler is easily entrapped by this local optimum. For this reason, the initial segmentation is usually chosen at random (Goldwater et al., 2009). Sentence-based block sampling is also susceptible to consistent initialization (Liang et al., 2010).

5.2 Type-based Sampling

To achieve fast convergence, we adopt a block sampling algorithm named type-based sampling (Liang et al., 2010). For the unigram model, a type-based sampler jointly samples multiple positions that share the same *type*. Two positions have the same type if the corresponding areas are both of the form w_1 or $w_2 w_3$. Type-based sampling takes advantage of the exchangeability of multiple positions with the same type. Given n positions with the same type, the sampler first samples the number of new boundaries m' ($0 \leq m' \leq n$), and then uniformly arranges m' boundaries out of n positions.

Type-based sampling has the ability to jump from a local optimum (e.g. consistently segmented) to another stable state (consistently unsegmented). While Liang et al. (2010) used random initialization, we take particular note of the possibility of efficiently correcting the consistent segmentation by the analyzer.

Type-based sampling is, however, not applicable to the bigram model for two reasons. The first problem is sparsity. For the bigram model, we need to consider adjacent words, w_l on the left and w_r on the right. This means that each type consists of three or four words, $w_l w_1 w_r$ or $w_l w_2 w_3 w_r$. Consequently, few positions share the same type and we fail to change closely-related areas $w_{l'} w_1 w_{r'}$ and $w_{l'} w_2 w_3 w_{r'}$, making inference inefficient.

The second and more fundamental problem arises from the hierarchical settings. Since the bigram model depends on latent table assignments, the joint distribution of multiple positions is no longer a closed-form function of counts.

Strictly speaking, we need to update the model counts even when sampling one position because the observation of the bigram $\langle w_l w_1 \rangle$, for example, may affect the probability $P_2(w_2 | \mathbf{h}_-, \langle w_l w_1 \rangle)$. Goldwater et al. (2009) approximate the probability by not updating the model counts in collapsed Gibbs

sampling (i.e. $P_2(w_2|\mathbf{h}_-, \langle w_l w_1 \rangle) \approx P_2(w_2|\mathbf{h}_-)$). They rely on the assumption that repeated bigrams are rare. Obviously this does not hold true for type-based sampling. Hence for type-based sampling, we have to update the model counts whenever we observe a new word.

One way to obtain the joint probability is to explicitly simulate the updates of histograms and other model counts. This is very cumbersome as we need to simulate $n + 1$ ways of model updates.

5.3 Hybrid Type-based Sampling

To address these problems, we propose a hybrid sampler which incorporates the Metropolis-Hastings algorithm into blocked Gibbs sampling. Metropolis-Hastings is another technique for sampling from a Markov chain. It first draws a proposed next state h' based on the current state h according to some proposal distribution $Q(h'; h)$. Then it accepts the proposal with the probability of

$$\min\left\{\frac{P(h')Q(h;h')}{P(h)Q(h';h)}, 1\right\}. \quad (4)$$

If the proposal is not accepted, the current state is used as the next state. Metropolis-Hastings is useful when it is difficult to directly sample from P.

We use the Metropolis-Hastings algorithm within Gibbs sampling. Instead of calculating the $n + 1$ probabilities of the number of boundaries, we only compare the current state with a proposed boundary arrangement. Also, the set of positions sampled jointly is chosen at unigram-level type instead of bigram-level type. The positions are no longer exchangeable. Therefore we calculate the conditional probability of one specific boundary arrangement.

When $n = 1$, the only choice is to flip the current state (i.e. $(m, m') \in \{(0, 1), (1, 0)\}$). This reduces to simple collapsed sampling. Otherwise we draw a proposed state in two steps. Given the n positions and the number of current boundaries m, we first draw the number of proposed boundaries m' from a probability distribution $f_n(m'; m)$. We then randomly arrange m' boundaries. The probability mass is uniformly divided by ${}_nC_{m'}$ arrangements. One exception is the case when $m \notin \{0, n\}$ and $m' = m$. In this case we perform permutation to obtain $h' \neq h$. To sum up, the proposal distribution

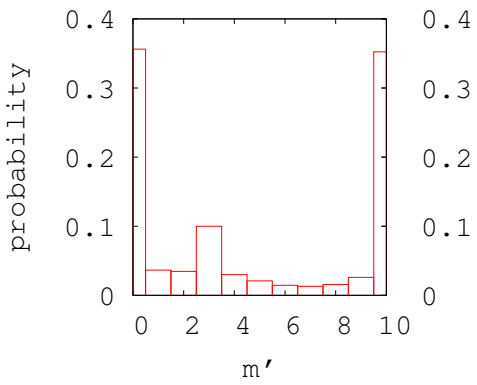


Figure 1: Probability of # of boundaries $f_{10}(m'; 3)$.

is defined as follows:

$$Q(h'; h) = \frac{f_n(m'; m)}{{}_nC_{m'} - I_n(m, m')}, \quad (5)$$

where $I_n(m, m')$ is 1 if $m \notin \{0, n\}$ and $m' = m$; otherwise 0.

We construct $f_n(m'; m)$ by discretizing a beta distribution ($\alpha = \beta < 1$) and a normal distribution with mean m, as shown in Figure 1. The former favors extreme values while the latter prefers smaller moves.

The sampling of each type is done in the following steps.

1. Collect n positions that share a unigram-level type.
2. Propose a new boundary arrangement. In what follows, we only focus on flipped boundaries because the rest does not change the likelihood ratio of the current and proposed states.
3. Calculate the current conditional probability. This can be done by repeatedly applying (2) while removing words one-by-one and updating the model counts accordingly.
4. Calculate the proposed conditional probability while adding words one-by-one.
5. Decide whether to accept the proposal according to (4). If the proposal is accepted, we finalize the arrangement; otherwise we revert to the current state.

We implement skip approximation (Liang et al., 2010) and sample each type once per iteration. This is motivated by the observation that although the

joint sampling of a large number of positions is computationally expensive, the proposal is accepted very infrequently.

5.4 Additional Constraints

Partial annotations (Tsuboi et al., 2008; Neubig and Mori, 2010) can be used for inference. If we know in advance that a certain position is a boundary or non-boundary, we simply keep it unaltered. As partially-annotated text, we can use markup. Suppose that the original text is written with wiki markup as follows:

*JR[[宇野線]][[常山駅]]
[gloss] JR Ube Line Tsuneyama Station

It is clear that the position between "線" (line) and "常" (*tsune*) is a boundary.

Similarly, we can impose our trivial rules of segmentation on the model. For example, we can keep punctuation markers (Li and Sun, 2009) separate from others.

6 Experiments

6.1 Settings

Data Set We evaluated our approach on Japanese Wikipedia. For each entry of Wikipedia, we regarded the title as a noun phrase and used both the title and main text for segmentation. We separately applied our segmentation procedure to each entry.

We constructed the data set as follows. We extracted each entry from an XML dump of Japanese Wikipedia.[4] We normalized the title by dropping trailing parentheses that disambiguate entries with similar names (e.g. "赤城 (空母)" for Akagi (aircraft carrier)). We extracted the main text from wikitext and used wiki markup as boundary markers. We applied both the title and main text to the morphological analyzer JUMAN[5] to get an initial segmentation. If the resultant segmentation conflicted with markup information, we overrode the former. The initial segmentation was also used as the baseline.

We only used entries that satisfied all of the following conditions.

1. The (normalized) title is longer than one character and contains hiragana, katakana and/or kanji.
2. The main text is longer than 1,000 characters.
3. The title appears at least 5 times in the main text.

The first condition ensures that there are segmentation ambiguities. The second and third conditions exclude entries unsuitable for statistical methods. 14% of the entries satisfied these conditions.

We randomly selected 500 entries and manually segmented their titles for evaluation. The 2-person inter-annotator Kappa score was 0.95.

As an annotated corpus, we used Kyoto Text Corpus.[6] It contained 1,675,188 characters.

Models We compared the unigram and bigram models. As for inference procedures, we used collapsed Gibbs sampling (**CL**) for both models, type-based sampling (**TB**) for the unigram model and hybrid type-based sampling (**HTB**) for the bigram model.

We tested two mixing methods of the annotated corpus, direct mixing (**DM**) and back-off mixing (**BM**).

To investigate the effect of initialization, we also tried randomly segmented text as the initial state (**RAND**). For random initialization, we placed a boundary with probability 0.5 on each position unless it was a fixed boundary.

The unigram model has one Dirichlet process concentration hyperparameter α_0 and the bigram model has α_0 and α_1. For each model, we experimented with the following values.
α_0: 0.1, 0.5, 1 5 10, 50, 100, 500, 1,000 and 5,000
α_1: 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100 and 500
For comparison, we also performed hyperparameter sampling. Following Escobar and West (1995), we set a gamma prior and introduced auxiliary variables to infer concentration parameters from data. For back-off mixing, we used the linear interpolation parameter $\lambda_{\mathrm{IP}} = 0.5$. The zerogram model was trained on the annotated corpus.

In each run, we performed 10 burn-in iterations. We then performed another 10 iterations to collect samples.

[4] http://download.wikimedia.org/jawiki/
[5] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN
[6] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?Kyoto%20University%20Text%20Corpus

Table 1: Results of segmentation of entry titles (F-score (precision/recall)).

model	best		median		inferred	
unigram + CL	81.35	(77.78/85.27)**	80.09	(75.80/84.89)	80.86	(76.81/85.36)
unigram + TB	55.87	(66.71/48.06)	51.04	(62.64/43.06)	42.63	(54.91/34.84)
bigram + CL	80.65	(76.73/84.99)	79.96	(75.50/84.99)	80.54	(76.84/84.61)
bigram + HTB	83.23	(85.25/81.30)**	74.52	(71.33/78.00)	34.52	(46.69/27.38)
unigram + CL + DM	85.29	(83.14/**87.54**)**	**81.62**	(**77.93**/**85.70**)**	80.91	(**82.87**/79.04)
unigram + TB + DM	35.26	(47.74/29.95)	33.81	(46.20/26.66)	31.90	(44.30/24.93)
bigram + CL + DM	80.37	(76.01/85.27)	79.88	(75.42/84.89)	73.77	(78.49/69.59)
bigram + HTB + DM	69.66	(67.68/71.77)	67.39	(64.35/70.73)	31.54	(43.79/24.64)
unigram + CL + BM	81.28	(77.48/85.46)	80.23	(76.06/84.89)	81.42	(77.75/**85.46**)
unigram + TB + BM	57.22	(68.01/49.39)	52.98	(64.50/44.95)	42.43	(54.69/34.66)
bigram + CL + BM	81.33	(77.34/85.74)	80.07	(75.69/84.99)	**81.46**	(77.82/**85.46**)**
bigram + HTB + BM	**86.32**	(**85.67**/86.97)**	76.35	(71.89/81.40)	40.81	(53.35/33.05)
unigram + TB + RAND	56.01	(66.93/48.16)	50.89	(62.21/43.06)	42.68	(54.81/34.94)
bigram + HTB + RAND	79.68	(80.13/79.23)	68.16	(63.64/73.37)	34.99	(47.05/27.86)
unigram + TB + BM + RAND	57.44	(67.91/49.76)	50.86	(61.92/43.15)	42.31	(54.55/34.56)
bigram + HTB + BM + RAND	84.03	(83.10/84.99)	70.46	(65.25/76.58)	40.16	(52.60/32.48)
baseline (JUMAN)	80.09	(75.80/84.89)				

** Statistically significant improvement with $p < 0.01$.

Evaluation Metrics We evaluated the segmentation accuracy of 500 entry titles. Specifically we evaluated the performance of a model with precision, recall and the F-score, all of which were based on tokens. We report the score of the most frequent segmentation among 10 samples.

Following Lee et al. (2010), we report the best and median settings of hyperparameters based on the F-score, in addition to inferred values.

In order to evaluate the degree of difference between a pair of segmentations, we employed character-based evaluation. Following Kudo et al. (2004), we converted a word sequence into character-based BI labels and examined labeling disagreements. McNemar's test of significance was based on this metric.

6.2 Results

Table 1 shows segmentation accuracy of various models. One would notice that the baseline score is much lower than the score previously reported regarding newspaper articles (Kudo et al., 2004). It is because unlike newspaper articles, the titles of Wikipedia entries contain an unusually high proportion of unknown words. As suggested by relatively low precision, unknown words tend to be over-segmented by the morphological analyzer.

In the best hyperparameter settings, the back-off mixing bigram model with hybrid type-based sampling (bigram + HTB + BM) significantly outperformed the baseline and achieved the best F-score. It did not performed well in the median setting as it was sensitive to the value of α_1. Hyperparameter estimation led to catastrophic decreases in bigram models as it made the hyperparameters much larger than those in the best settings.

Collapsed sampling (+CL) returned scores comparable to that of the baseline. It is simply because it did not change the initial segmentation a lot. In contrast, type-based sampling (+TB) brought large moves to the unigram model and significantly hurt accuracy. As suggested by relatively low recall, the unigram model prefers under-segmentation.

When combined with (hybrid) type-based sampling (+TB/+HTB), back-off mixing (+BM) increased accuracy from the corresponding non-mixing models. By contrast, direct mixing (+DM) drastically decreased accuracy from the non-mixing models. We can confirm that when the main text is orders of magnitude smaller than the annotated text, the role of constituent words in the main text is underestimated. To our surprise, collapsed sampling with mixing models (+CL, +DM/+BM) outperformed the baseline. However, the scores of type-based sampling (+TB) suggest that with much more iterations, the models would converge to undesired states.

The unigram model with random initialization was indifferent from that with default initialization. By contrast, the performance of the bigram model slightly degenerated with random initialization.

6.3 Convergence

Figure 2 shows how segmentations differed from the initial state in the course of inference.[7] A *diff* is defined as the number of character-based disagreements between the baseline segmentation and a model output. Hyperparameters used were those of the best model with (hybrid) type-based sampling.

We can see that collapsed sampling was almost unable to escape the initial state. With type-based sampling (+TB), the unigram model went further than the bigram model, but to an undesired direction. The bigram model with hybrid type-based sampling (bigram + HTB) converged in few iterations. Although the model with random initialization (+RAND) converged to a nearby point, the initial segmentation by the morphological analyzer realized a bit faster convergence and better accuracy.

Figure 2 shows how acceptance rates changed during inference. For comparison, a sample by a type-based Gibbs sampler was treated as "accepted" if the number of new boundaries was different from that of the current boundaries (i.e. $m' \neq m$). The acceptance rates were low and samplers seemingly stayed around modes.

6.4 Approximation

Up to this point, we consider every possible boundary position. However, this seems wasteful, given that a large portion of text has only marginal influence on the segmentation of the noun phrase in question. For this reason, we implemented approximation named matching skip. We sampled a boundary only if the corresponding local area contained a substring of the noun phrase in question.

Table 2 shows the result of approximation. Hyperparameters used were those of the best models with full sampling. Matching skip steadily worsened performance although not to a large extent. Meanwhile it drastically reduced the number of sampled positions. The median skip rate was 90.87%, with a standard deviation of 8.5.

[7] For a fair comparison, we might need to report changes over time instead of iterations. However, the difference of convergence speed is obvious in the iteration-based comparison although (hybrid) type-based sampling takes several times longer than collapsed sampling in the current naïve implementation.

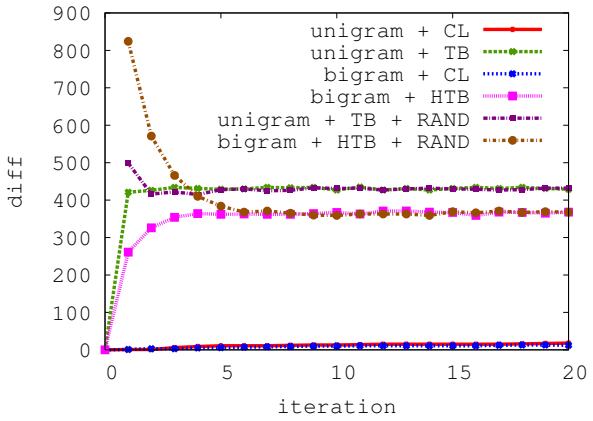


Figure 2: Diffs in the course of iteration. All models were with back-off mixing (+BM).

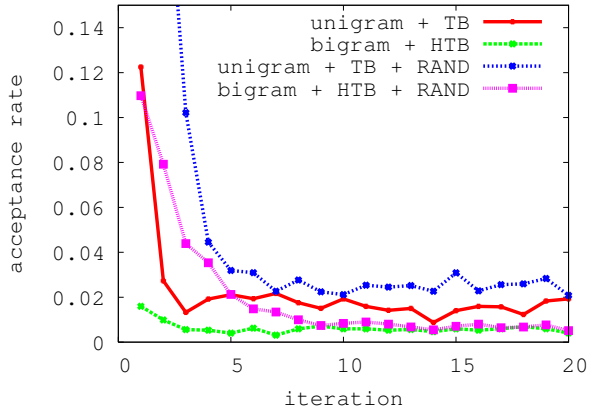


Figure 3: Acceptance rates for a noun phrase in the course of iteration. All models were with back-off mixing (+BM).

6.5 Discussion

Figure 4 shows some segmentations corrected by the back-off mixing bigram model with hybrid type-based sampling. "市比野" (*ichihino*) is a rare place name but can be identified by the model because it is frequently used in the article. "こなみるく" (*konamiruku* in hiragana) seems a pun on "粉ミルク" (*kona miruku*, "powdered milk") and "コナミ" (*konami* in katakana, a company). We consider it as a single word because we cannot reconstruct the etymology solely based on the main text. Note the different scripts. In Japanese, people often change the script to derive a proper noun from a common noun, which a naïve analyzer fails to recognize. It is

Table 2: Effect of matching skip (F-score (precision/recall)).

model	full		matching skip	
bigram + HTB	83.23	(85.25/81.30)**	82.86	(84.27/81.49)
bigram + HTB + BM	86.32	(85.67/86.97)**	83.87	(82.60/85.17)**
bigram + HTB + RAND	79.68	(80.13/79.23)	78.81	(78.64/75.07)
bigram + HTB + BM + RAND	84.03	(83.10/84.99)	81.08	(80.22/81.96)
baseline (JUMAN)	80.09	(75.80/84.89)		

** Statistically significant improvement with $p < 0.01$.

樋 + 脇 + 町 + 市 + 比 + 野 ⇒ 樋脇 + 町 + 市比野
hiwaki chou ichihino
(Ichihino, Hiwaki Town, an address)

り + そな + カード ⇒ りそな + カード
risona kaRdo
(Risona Card, a company)

ちり + とて + ちん ⇒ ちりとてちん
chiritotechiN (name of a play)

こな + みる + く ⇒ こなみるく
konamiruku
(a shop affiliated with Konami Corporation)

はい + じぃ ⇒ はいじぃ
haiziI (stage name of a comedian)

ちん + すこう ⇒ ちんすこう
chiNsukou (a traditional sweet)

コントラアルトクラリネット ⇒ コントラ + アルト + クラリネット
koNtora aruto kurarineQto (Contra-alto clarinet)

Figure 4: Examples of improved segmentations.

very important to identify hiragana words correctly. As hiragana is mainly used to write function words and other basic words, segmentation errors concerning hiragana often bring disastrous effects on applications of morphological analysis. For example, the analyzer over-segments "ちりとてちん" (*chiritotechiN*) into three shorter words among which the second word "とて" (*tote*) is a particle, and this sequence of words is transformed into a terrible parse tree.

Most improvements come from correction of over-segmentation because the initial segmentation by the analyzer shows a tendency of over-segmentation. An example of corrected under-segmentation is "contra-alto clarinet." The presence of "clarinet," "alto" and "contrabass" and others in the main text allowed the model to identify the constituents. On the other hand, the segmentation failed when our assumption about constituents does not hold. For example, the person name "菊池俊吉" (*kikuchi shuNkichi*) is two words but was erroneously combined into a single word by the model because unfortunately he was always referred to by the full name.

7 Conclusions

In this paper, we proposed a new task of Japanese noun phrase segmentation. We adopted non-parametric Bayesian language models and proposed hybrid type-based sampling that can efficiently correct segmentation given by the morphological analyzer. Although supervised segmentation is very competitive, we showed that it can be supplemented with our unsupervised approach.

We applied the proposed method to encyclopedic text to segment noun phrases in it. The proposed method can be applied to other tasks. For example, in unknown word acquisition (Murawaki and Kurohashi, 2008), noun phrases are often acquired from text as single words. We can now segment them into words in a more sophisticated way.

In the future we will assign a POS tag to each word in order to use segmented noun phrases in morphological analysis. We assume that the meaning of constituents in a noun phrase rarely depends on outer context. So it would be helpful to augment them with rich semantic information in advance instead of disambiguating their meaning every time we analyze given text.

Acknowledgments

This work was partly supported by JST CREST.

References

Masayuki Asahara and Yuji Matsumoto. 2000. Extended models and tools for high-performance part-of-speech

tagger. In *Proc. of COLING 2000*, pages 21–27.

Masayuki Asahara and Yuji Matsumoto. 2004. Japanese unknown word identification by character-based chunking. In *Proc. COLING 2004*, pages 459–465.

Phil Blunsom, Trevor Cohn, Sharon Goldwater, and Mark Johnson. 2009. A note on the implementation of hierarchical Dirichlet processes. In *Proc. of ACL-IJCNLP 2009: Short Papers*, pages 337–340.

Michael D. Escobar and Mike West. 1995. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588.

Thomas S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In *NIPS 18*, pages 459–466.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Mark Johnson and Katherine Demuth. 2010. Unsupervised phonemic Chinese word segmentation using adaptor grammars. In *Proc. of COLING 2010*, pages 528–536.

Mark Johnson. 2008. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proc. of ACL 2008*, pages 398–406.

Jun'ichi Kazama and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *Proc. of ACL 2008*, pages 407–415, June.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proc. of EMNLP 2004*, pages 230–237.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proc. of The International Workshop on Sharable Natural Language Resources*, pages 22–38.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2010. Simple type-level unsupervised POS tagging. In *Proc. of EMNLP 2010*, pages 853–861.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for Chinese word segmentation. *Computational Linguistics*, 35(4):505–512.

Percy Liang, Michael I. Jordan, and Dan Klein. 2010. Type-based MCMC. In *Proc. of NAACL 2010*, pages 573–581.

Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. 2005. A maximum entropy approach to Chinese word segmentation. In *Proc. of the 4th SIGHAN Workshop*, pages 161–164.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proc. of ACL-IJCNLP 2009*, pages 100–108.

Yugo Murawaki and Sadao Kurohashi. 2008. Online acquisition of Japanese unknown morphemes using morphological constraints. In *Proc. of EMNLP 2008*, pages 429–437.

Masaaki Nagata. 1996. Automatic extraction of new words from Japanese texts using generalized forward-backward search. In *Proc. of EMNLP 1996*, pages 48–59.

Tetsuji Nakagawa and Yuji Matsumoto. 2006. Guessing parts-of-speech of unknown words using global information. In *Proc. of COLING-ACL 2006*, pages 705–712.

Graham Neubig and Shinsuke Mori. 2010. Word-based partial annotation for efficient corpus construction. In *Proc. of LREC 2010*.

ThuyLinh Nguyen, Stephan Vogel, and Noah A. Smith. 2010. Nonparametric word segmentation for machine translation. In *Proc. of COLING 2010*, pages 815–823.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proc. of COLING '04*, pages 562–568.

Yee Whye Teh. 2006. A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, National University of Singapore.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proc. of COLING 2008*, pages 897–904.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 2001. The unknown word problem: a morphological analysis of Japanese using maximum entropy aided by a dictionary. In *Proc. of EMNLP 2001*, pages 91–99.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised Chinese word segmentation for statistical machine translation. In *Proc. of COLING 2008*, pages 1017–1024.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Toshio Yokoi. 1995. The EDR electronic dictionary. *Communications of the ACM*, 38(11):42–44.

Discovering Morphological Paradigms from Plain Text Using a Dirichlet Process Mixture Model

Markus Dreyer*
SDL Language Weaver
Los Angeles, CA 90045, USA
mdreyer@sdl.com

Jason Eisner
Computer Science Dept., Johns Hopkins University
Baltimore, MD 21218, USA
jason@cs.jhu.edu

Abstract

We present an inference algorithm that organizes observed words (tokens) into structured inflectional paradigms (types). It also naturally predicts the spelling of unobserved forms that are missing from these paradigms, and discovers inflectional principles (grammar) that generalize to wholly unobserved words.

Our Bayesian generative model of the data explicitly represents tokens, types, inflections, paradigms, and locally conditioned string edits. It assumes that inflected word tokens are generated from an infinite mixture of inflectional paradigms (string tuples). Each paradigm is sampled all at once from a graphical model, whose potential functions are weighted finite-state transducers with language-specific parameters to be learned. These assumptions naturally lead to an elegant empirical Bayes inference procedure that exploits Monte Carlo EM, belief propagation, and dynamic programming. Given 50–100 seed paradigms, adding a 10-million-word corpus reduces prediction error for morphological inflections by up to 10%.

1 Introduction

1.1 Motivation

Statistical NLP can be difficult for morphologically rich languages. Morphological transformations on words increase the size of the observed vocabulary, which unfortunately masks important generalizations. In Polish, for example, each lexical verb has literally 100 inflected forms (Janecki, 2000). That is, a single *lexeme* may be realized in a corpus as many different word types, which are differently inflected for person, number, gender, tense, mood, etc.

All this makes lexical features even sparser than they would be otherwise. In machine translation or text generation, it is difficult to learn *separately* how to translate, or when to generate, each of these many word types. In text analysis, it is difficult to learn lexical features (as cues to predict topic, syntax, semantics, or the next word), because one must learn a separate feature for each word form, rather than generalizing across inflections.

Our engineering goal is to address these problems by mostly-unsupervised learning of morphology. Our linguistic goal is to build a generative probabilistic model that directly captures the basic representations and relationships assumed by morphologists. This model suffices to *define* a posterior distribution over analyses of any given collection of type and/or token data. Thus we obtain scientific data interpretation as probabilistic inference (Jaynes, 2003). Our computational goal is to *estimate* this posterior distribution.

1.2 What is Estimated

Our inference algorithm jointly reconstructs *token*, *type*, and *grammar* information about a language's morphology. This has not previously been attempted.

Tokens: We will tag each word token in a corpus with (1) a *part-of-speech (POS) tag*,[1] (2) an *inflection*, and (3) a *lexeme*. A token of `broken` might be tagged as (1) a VERB and more specifically as (2) the past participle inflection of (3) the abstract lexeme *break*.[2]

Reconstructing the latent lexemes and inflections allows the features of other statistical models to consider them. A parser may care that `broken` is a past participle; a search engine or question answering system may care that it is a form of *break*; and a translation system may care about both facts.

* This research was done at Johns Hopkins University as part of the first author's dissertation work. It was supported by the Human Language Technology Center of Excellence and by the National Science Foundation under Grant No. 0347822.

[1] POS tagging may be done as part of our Bayesian model or beforehand, as a preprocessing step. Our experiments chose the latter option, and then analyzed only the verbs (see section 8).

[2] We use cursive font for abstract lexemes to emphasize that they are atomic objects that do not decompose into letters.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 616–627,
Edinburgh, Scotland, UK, July 27–31, 2011.

		singular	plural
present	1st-person	breche	brechen
	2nd-person	brichst	brecht
	3rd-person	bricht	brechen
past	1st-person	brach	brachen
	2nd-person	brachst	bracht
	3rd-person	brach	brachen

Table 1: Part of a morphological paradigm in German, showing the spellings of some inflections of the lexeme *break* (whose lemma is brechen), organized in a grid.

Types: In carrying out the above, we will reconstruct specific *morphological paradigms* of the language. A paradigm is a grid of all the inflected forms of some lexeme, as illustrated in Table 1. Our reconstructed paradigms will include our predictions of inflected forms that were never observed in the corpus. This tabular information about the types (rather than the tokens) of the language may be separately useful, for example in translation and other generation tasks, and we will evaluate its accuracy.

Grammar: We estimate *parameters* $\vec{\theta}$ that describe general patterns in the language. We learn a prior distribution over inflectional paradigms by learning (e.g.) how a verb's suffix or stem vowel tends to change when it is pluralized. We also learn (e.g.) whether singular or plural forms are more common. Our basic strategy is Monte Carlo EM, so these parameters tell us how to guess the paradigms (Monte Carlo E step), then these reconstructed paradigms tell us how to reestimate the parameters (M step), and so on iteratively. We use a few supervised paradigms to initialize the parameters and help reestimate them.

2 Overview of the Model

We begin by sketching the main ideas of our model, first reviewing components that we introduced in earlier papers. Sections 5–7 will give more formal details. Full details and more discussion can be found in the first author's dissertation (Dreyer, 2011).

2.1 Modeling Morphological Alternations

We begin with a family of joint distributions $p(x, y)$ over string pairs, parameterized by $\vec{\theta}$. For example, to model just the semi-systematic relation between a German lemma and its 3rd-person singular present form, one could train $\vec{\theta}$ to maximize the likelihood of (x, y) pairs such as (brechen, bricht). Then, given a lemma x, one could predict its inflected form y via $p(y \mid x)$, and vice-versa.

Dreyer et al. (2008) define such a family via a log-linear model with latent alignments,

$$p(x, y) = \sum_a p(x, y, a) \propto \sum_a \exp(\vec{\theta} \cdot \vec{f}(x, y, a))$$

Here a ranges over monotonic 1-to-1 character alignments between x and y. $\propto$ means "proportional to" (p is normalized to sum to 1). $\vec{f}$ extracts a vector of local features from the aligned pair by examining trigram windows. Thus $\vec{\theta}$ can reward or penalize specific features—e.g., insertions, deletions, or substitutions in specific contexts, as well as trigram features of x and y separately.[3] Inference and training are done by dynamic programming on finite-state transducers.

2.2 Modeling Morphological Paradigms

A paradigm such as Table 1 describes how some abstract lexeme (*break*) is *expressed* in German.[4] We evaluate *whole paradigms* as linguistic objects, following word-and-paradigm or realizational morphology (Matthews, 1972; Stump, 2001). That is, we presume that some language-specific distribution $p(\pi)$ defines whether a paradigm π is a grammatical—and *a priori* likely—way for a lexeme to express itself in the language. Learning $p(\pi)$ helps us reconstruct paradigms, as described at the end of section 1.2.

Let $\pi = (x_1, x_2, \ldots)$. In Dreyer and Eisner (2009), we showed how to model $p(\pi)$ as a renormalized *product* of many pairwise distributions $p_{rs}(x_r, x_s)$, each having the log-linear form of section 2.1:

$$p(\pi) \propto \prod_{r,s} p_{rs}(x_r, x_s) \propto \exp(\sum_{r,s} \vec{\theta} \cdot \overrightarrow{f_{rs}}(x_r, x_s, a_{rs}))$$

This is an undirected graphical model (MRF) over *string-valued* random variables x_s; each factor p_{rs} evaluates the relationship between some pair of strings. Note that it is still a log-linear model, and parameters in $\vec{\theta}$ can be reused across different rs pairs.

To guess at unknown strings in the paradigm, Dreyer and Eisner (2009) show how to perform approximate inference on such an MRF by loopy belief

[3] Dreyer et al. (2008) devise additional helpful features based on enriching the aligned pair with additional latent information, but our present experiments drop those for speed.

[4] Our present experiments focus on orthographic forms, because we are learning from a written corpus. But it would be natural to use phonological forms instead, or to include both in the paradigm so as to model their interrelationships.

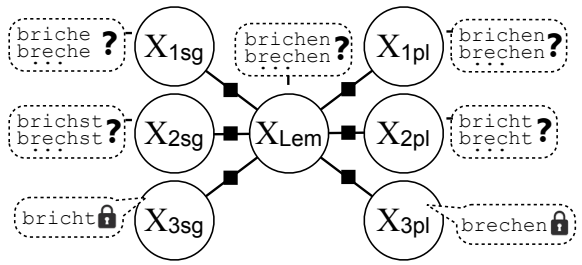


Figure 1: A distribution over paradigms modeled as an MRF over 7 strings. Random variables X_{Lem}, X_{1st}, etc., are the lemma, the 1st person form, etc. Suppose two forms are observed (denoted by the "lock" icon). Given these observations, belief propagation estimates the posterior marginals over the other variables (denoted by "?").

propagation, using finite-state operations. It is not necessary to include all rs pairs. For example, Fig. 1 illustrates the result of belief propagation on a simple MRF whose factors relate all inflected forms to a common (possibly unobserved) lemma, but not directly to one another.[5]

Our method could be used with any $p(\pi)$. To speed up inference (see footnote 7), our present experiments actually use the *directed* graphical model variant of Fig. 1—that is, $p(\pi) = p_1(x_1) \cdot \prod_{s>1} p_{1s}(x_s \mid x_1)$, where x_1 denotes the lemma.

2.3 Modeling the Lexicon (types)

Dreyer and Eisner (2009) learned $\vec{\theta}$ by partially observing some paradigms (type data). That work, while rather accurate at predicting inflected forms, sometimes erred: it predicted spellings that never occurred in text, even for forms that "should" be common. To fix this, we shall incorporate an unlabeled or POS-tagged corpus (token data) into learning.

We therefore need a model for generating tokens—a *probabilistic lexicon* that specifies which inflections of which lexemes are common, and how they are spelled. We do not know our language's probabilistic lexicon, but we assume it was generated as follows:

1. Choose parameters $\vec{\theta}$ of the MRF. This defines $p(\pi)$: which paradigms are likely *a priori*.
2. Choose a distribution over the abstract lexemes.
3. For each lexeme, choose a distribution over its inflections.
4. For each lexeme, choose a paradigm that will be used to express the lexeme orthographically.

Details are given later. Briefly, step 1 samples $\vec{\theta}$ from a Gaussian prior. Step 2 samples a distribution from a Dirichlet process. This chooses a countable number of lexemes to have positive probability in the language, and decides which ones are most common. Step 3 samples a distribution from a Dirichlet. For the lexeme *think*, this might choose to make 1st-person singular more common than for typical verbs. Step 4 just samples IID from $p(\pi)$.

In our model, each part of speech generates its own lexicon: VERBs are inflected differently from NOUNs (different parameters and number of inflections). The size and layout of (e.g.) VERB paradigms is language-specific; we currently assume it is given by a linguist, along with a few supervised VERB paradigms.

2.4 Modeling the Corpus (tokens)

At present, we use only a very simple exchangeable model of the corpus. We assume that each word was independently sampled from the lexicon given its part of speech, with no other attention to context.

For example, a token of `brechen` may have been chosen by choosing frequent lexeme *break* from the VERB lexicon; then choosing 1st-person plural given *break*; and finally looking up that inflection's spelling in *break*'s paradigm. This final lookup is deterministic since the lexicon has already been generated.

3 A Sketch of Inference and Learning

3.1 Gibbs Sampling Over the Corpus

Our job in inference is to reconstruct the lexicon that was used and how each token was generated from it (i.e., which lexeme and inflection?). We use collapsed Gibbs sampling, repeatedly guessing a reanalysis of each token in the context of all others. Gradually, similar tokens get "clustered" into paradigms (section 4).

The state of the sampler is illustrated in Fig. 2. The bottom half shows the current analyses of the verb tokens. Each is associated with a particular slot in some paradigm. We are now trying to reanalyze `brechen` at position ❼. The dashed arrows show some possible analyses.

[5] This view is adopted by some morphological theorists (Albright, 2002; Chan, 2006), although see Appendix E.2 for a caution about syncretism. Note that when the lemma is unobserved, the other forms do still influence one another indirectly.

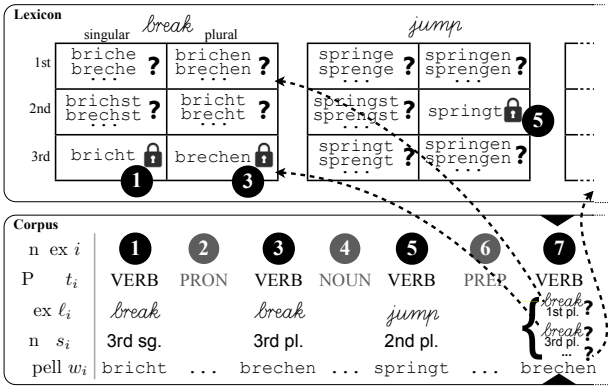


Figure 2: A state of the Gibbs sampler (note that the assumed generative process runs roughly top-to-bottom). Each corpus token i has been tagged with part of speech t_i, lexeme ℓ_i and inflection s_i. Token ❶ has been tagged as *break* and 3rd sg., which locked the corresponding type spelling in the paradigm to the spelling $w_1 =$ `bricht`; similarly for ❸ and ❺. Now w_7 is about to be reanalyzed.

The key intuition is that the current analyses of the *other* verb tokens imply a posterior distribution over the VERB lexicon, shown in the top half of the figure.

First, because of the current analyses of ❶ and ❸, the 3rd-person spellings of *break* are already constrained to match w_1 and w_3 (the "lock" icon).

Second, belief propagation as in Fig. 1 tells us which other inflections of *break* (the "?" icon) are plausibly spelled as `brechen`, and how likely they are to be spelled that way.

Finally, the fact that other tokens are associated with *break* suggest that this is a popular lexeme, making it a plausible explanation of ❼ as well. (This is the "rich get richer" property of the Chinese restaurant process; see section 6.6.) Furthermore, certain inflections of *break* appear to be especially popular.

In short, given the other analyses, we know which inflected lexemes in the lexicon are likely, *and* how likely each one is to be spelled as `brechen`. This lets us compute the relative probabilities of the possible analyses of token ❼, so that the Gibbs sampler can accordingly choose one of these analyses at random.

3.2 Monte Carlo EM Training of $\vec{\theta}$

For a given $\vec{\theta}$, this Gibbs sampler converges to the posterior distribution over analyses of the full corpus. To improve our $\vec{\theta}$ estimate, we periodically adjust $\vec{\theta}$ to maximize or increase the probability of the most recent sample(s). For example, having tagged $w_5 =$ `springt` as $s_5 =$ 2nd-person plural may strengthen our estimated probability that 2nd-person spellings tend to end in `-t`. That revision to $\vec{\theta}$, in turn, will influence future moves of the sampler.

If the sampler is run long enough between calls to the $\vec{\theta}$ optimizer, this is a Monte Carlo EM procedure (see end of section 1.2). It uses the data to optimize a language-specific prior $p(\pi)$ over paradigms—an empirical Bayes approach. (A fully Bayesian approach would resample $\vec{\theta}$ as part of the Gibbs sampler.)

3.3 Collapsed Representation of the Lexicon

The lexicon is *collapsed out* of our sampler, in the sense that we do not represent a single guess about the infinitely many lexeme probabilities and paradigms. What we store about the lexicon is information about its full posterior distribution: the top half of Fig. 2.

Fig. 2 names its lexemes as *break* and *jump* for expository purposes, but of course the sampler cannot reconstruct such labels. Formally, these labels are collapsed out, and we represent lexemes as anonymous objects. Tokens ❶ and ❸ are tagged with the *same* anonymous lexeme (which will correspond to sitting at the same table in a Chinese restaurant process).

For each lexeme ℓ and inflection s, we maintain pointers to any tokens currently tagged with the slot (ℓ, s). We also maintain an approximate marginal distribution over the spelling of that slot:[6]

1. If (ℓ, s) points to at least one token i, then we know (ℓ, s) is spelled as w_i (with probability 1).

2. Otherwise, the spelling of (ℓ, s) is not known. But if some spellings in ℓ's paradigm are known, store a truncated distribution that enumerates the 25 most likely spellings for (ℓ, s), according to loopy belief propagation within the paradigm.

3. Otherwise, we have observed nothing about ℓ: it is currently unused. All such ℓ share the same marginal distribution over spellings of (ℓ, s): the marginal of the prior $p(\pi)$. Here a 25-best list could not cover all plausible spellings. Instead we store a probabilistic finite-state language model that approximates this marginal.[7]

[6] Cases 1 and 2 below must in general be *updated* whenever a slot switches between having 0 and more than 0 tokens. Cases 2 and 3 must be updated when the parameters $\vec{\theta}$ change.

[7] This character trigram model is fast to build if $p(\pi)$ is de-

A hash table based on cases 1 and 2 can now be used to rapidly map any word w to a list of slots of existing lexemes that might plausibly have generated w. To ask whether w might instead be an inflection s of a novel lexeme, we score w using the probabilistic finite-state automata from case 3, one for each s.

The Gibbs sampler randomly chooses one of these analyses. If it chooses the "novel lexeme" option, we create an arbitrary new lexeme object in memory. The number of explicitly represented lexemes is always finite (at most the number of corpus tokens).

4 Interpretation as a Mixture Model

It is common to cluster points in $\mathbb{R}^n$ by assuming that they were generated from a *mixture of Gaussians*, and trying to reconstruct which points were generated from the same Gaussian.

We are similarly clustering word tokens by assuming that they are generated from a *mixture of weighted paradigms*. After all, each word token was obtained by randomly sampling a weighted paradigm (i.e., a cluster) and then randomly sampling a word from it.

Just as each Gaussian in a Gaussian mixture is a distribution over all points $\mathbb{R}^n$, each weighted paradigm is a distribution over all spellings Σ^* (but assigns probability > 0 to only a finite subset of Σ^*).

Inference under our model clusters words together by tagging them with the same lexeme. It tends to group words that are "similar" in the sense that the base distribution $p(\pi)$ predicts that they would tend to co-occur within a paradigm. Suppose a corpus contains several unlikely but similar tokens, such as `discombobulated` and `discombobulating`. A language might have one probable lexeme from whose paradigm all these words were sampled. It is much less likely to have several probable lexemes that all *coincidentally* chose spellings that started with `discombobulat-`. Generating `discombobulat-` only once is cheaper (especially for such a long prefix), so the former explanation has higher probability. This is like explaining nearby points in $\mathbb{R}^n$ as samples from the same Gaussian. Of course, our model is sensitive to more than shared prefixes, and it does not merely cluster words into a paradigm but assigns them to particular inflectional slots in the paradigm.

fined as at the end of section 2.2. If not, one could still try belief propagation; or one could approximate by estimating a language model from the spellings associated with slot s by cases 1 and 2.

4.1 The Dirichlet Process Mixture Model

Our mixture model uses an *infinite* number of mixture components. This avoids placing a prior bound on the number of lexemes or paradigms in the language. We assume that a natural language has an infinite lexicon, although most lexemes have sufficiently low probability that they have not been used in our training corpus or even in human history (yet).

Our specific approach corresponds to a Bayesian technique, the Dirichlet process mixture model. Appendix A (supplementary material) explains the DPMM and discusses it in our context.

The DPMM would standardly be presented as generating a distribution over countably many Gaussians or paradigms. Our variant in section 2.3 instead broke this into two steps: it first generated a distribution over countably many lexemes (step 2), and then generated a weighted paradigm for each lexeme (steps 3–4). This construction keeps distinct lexemes separate even if they happen to have identical paradigms (polysemy). See Appendix A for a full discussion.

5 Formal Notation

5.1 Value Types

We now describe our probability model in more formal detail. It considers the following types of mathematical objects. (We use consistent lowercase letters for values of these types, and consistent fonts for constants of these types.)

A **word** w, such as `broken`, is a finite string of any length, over some finite, given alphabet Σ.

A **part-of-speech tag** t, such as VERB, is an element of a certain finite set $\mathcal{T}$, which in this paper we assume to be given.

An **inflection** s,[8] such as past participle, is an element of a finite set $\mathcal{S}_t$. A token's part-of-speech tag $t \in \mathcal{T}$ determines its set $\mathcal{S}_t$ of possible inflections. For tags that do not inflect, $|\mathcal{S}_t| = 1$. The sets $\mathcal{S}_t$ are language-specific, and we assume in this paper that they are given by a linguist rather than learned. A linguist also specifies features of the inflections: the grid layout in Table 1 shows that 4 of the 12 inflections in $\mathcal{S}_{\text{VERB}}$ share the "2nd-person" feature.

[8] We denote inflections by s because they represent "slots" in paradigms (or, in the metaphor of section 6.7, "seats" at tables in a Chinese restaurant). These slots (or seats) are filled by words.

A **paradigm** for $t \in \mathcal{T}$ is a mapping $\pi : \mathcal{S}_t \to \Sigma^*$, specifying a **spelling** for each inflection in $\mathcal{S}_t$. Table 1 shows one VERB paradigm.

A **lexeme** ℓ is an abstract element of some lexical space $\mathcal{L}$. Lexemes have no internal semantic structure: the only question we can ask about a lexeme is whether it is equal to some other lexeme. There is no upper bound on how many lexemes can be discovered in a text corpus; $\mathcal{L}$ is infinite.

5.2 Random Quantities

Our generative model of the corpus is a joint probability distribution over a collection of random variables. We describe them in the same order as section 1.2.

Tokens: The corpus is represented by *token* variables. In our setting the sequence of words $\vec{w} = w_1, \ldots, w_n \in \Sigma^*$ is observed, along with n. We must recover the corresponding part-of-speech tags $\vec{t} = t_1, \ldots, t_n \in \mathcal{T}$, lexemes $\vec{\ell} = \ell_1, \ldots, \ell_n \in \mathcal{L}$, and inflections $\vec{s} = s_1, \ldots, s_n$, where $(\forall i) s_i \in \mathcal{S}_{t_i}$.

Types: The lexicon is represented by *type* variables. For each of the infinitely many lexemes $\ell \in \mathcal{L}$, and each $t \in \mathcal{T}$, the paradigm $\pi_{t,\ell}$ is a function $\mathcal{S}_t \to \Sigma^*$. For example, Table 1 shows a possible value $\pi_{\text{VERB},\textit{break}}$. The various spellings in the paradigm, such as $\pi_{\text{VERB},\textit{break}}(\text{1st-person sing. pres.}) = \texttt{breche}$, are string-valued random variables that are correlated with one another.

Since the lexicon is to be probabilistic (section 2.3), $G_t(\ell)$ denotes tag t's distribution over lexemes $\ell \in \mathcal{L}$, while $H_{t,\ell}(s)$ denotes the tagged lexeme (t, ℓ)'s distribution over inflections $s \in \mathcal{S}_t$.

Grammar: Global properties of the language are captured by *grammar* variables that cut across lexical entries: our parameters $\vec{\theta}$ that describe typical inflectional alternations, plus parameters $\vec{\phi_t}, \alpha_t, \alpha'_t, \vec{\tau}$ (explained below). Their values control the overall shape of the probabilistic lexicon that is generated.

6 The Formal Generative Model

We now fully describe the generative process that was sketched in section 2. Step by step, it randomly chooses an assignment to all the random variables of section 5.2. Thus, a given assignment's probability—which section 3's algorithms consult in order to resample or improve the current assignment—is the product of the probabilities of the individual choices, as described in the sections below. (Appendix B provides a drawing of this as a graphical model.)

6.1 Grammar Variables $p(\vec{\theta}), p(\vec{\phi_t}), p(\alpha_t), p(\alpha'_t)$

First select the grammar variables from a prior. (We will see below how these variables get used.) Our experiments used fairly flat priors. Each weight in $\vec{\theta}$ or $\vec{\phi_t}$ is drawn IID from $\mathcal{N}(0, 10)$, and each α_t or α'_t from a Gamma with mode 10 and variance 1000.

6.2 Paradigms $p(\pi_{t,\ell} \mid \vec{\theta})$

For each $t \in \mathcal{T}$, let $D_t(\pi)$ denote the distribution over paradigms that was presented in section 2.2 (where it was called $p(\pi)$). D_t is fully specified by our graphical model for paradigms of part of speech t, together with its parameters $\vec{\theta}$ as generated above.

This is the linguistic core of our model. It considers spellings: D_{VERB} describes what verb paradigms typically look like in the language (e.g., Table 1).

Parameters in $\vec{\theta}$ may be shared across parts of speech t. These "backoff" parameters capture general phonotactics of the language, such as prohibited letter bigrams or plausible vowel changes.

For each possible tagged lexeme (t, ℓ), we now draw a paradigm $\pi_{t,\ell}$ from D_t. Most of these lexemes will end up having probability 0 in the language.

6.3 Lexical Distributions $p(G_t \mid \alpha_t)$

We now formalize section 2.3. For each $t \in \mathcal{T}$, the language has a distribution $G_t(\ell)$ over lexemes. We draw G_t from a Dirichlet process $\text{DP}(G, \alpha_t)$, where G is the **base distribution** over $\mathcal{L}$, and $\alpha_t > 0$ is a **concentration parameter** generated above. If α_t is small, then G_t will tend to have the property that most of its probability mass falls on relatively few of the lexemes in $\mathcal{L}_t \stackrel{\text{def}}{=} \{\ell \in \mathcal{L} : G_t(\ell) > 0\}$. A *closed-class tag* is one whose α_t is especially small.

For G to be a uniform distribution over an infinite lexeme set $\mathcal{L}$, we need $\mathcal{L}$ to be uncountable.[9] However, it turns out[10] that with probability 1, each $\mathcal{L}_t$ is *countably* infinite, and all the $\mathcal{L}_t$ are disjoint. So each lexeme $\ell \in \mathcal{L}$ is selected by at most one tag t.

[9] For example, $\mathcal{L} \stackrel{\text{def}}{=} [0, 1]$, so that *break* is merely a suggestive nickname for a lexeme such as 0.2538159.

[10] This can be seen by considering the stick-breaking construction of the Dirichlet process that (Sethuraman, 1994; Teh et al., 2006). A separate stick is broken for each G_t. See Appendix A.

6.4 Inflectional Distributions $p(H_{t,\ell} \mid \overrightarrow{\phi_t}, \alpha'_t)$

For each tagged lexeme (t, ℓ), the language specifies some distribution $H_{t,\ell}$ over its inflections.

First we construct backoff distributions H_t that are independent of ℓ. For each tag $t \in \mathcal{T}$, let H_t be some base distribution over $\mathcal{S}_t$. As $\mathcal{S}_t$ could be large in some languages, we exploit its grid structure (Table 1) to reduce the number of parameters of H_t. We take H_t to be a log-linear distribution with parameters $\overrightarrow{\phi_t}$ that refer to *features* of inflections. E.g., the 2nd-person inflections might be *systematically* rare.

Now we model each $H_{t,\ell}$ as an independent draw from a finite-dimensional Dirichlet distribution with mean H_t and concentration parameter α'_t. E.g., *think* might be biased toward 1st-person sing. present.

6.5 Part-of-Speech Tag Sequence $p(\vec{t} \mid \vec{\tau})$

In our current experiments, $\vec{t}$ is given. But in general, to discover tags and inflections simultaneously, we can suppose that the tag sequence $\vec{t}$ (and its length n) are generated by a Markov model, with tag bigram or trigram probabilities specified by some parameters $\vec{\tau}$.

6.6 Lexemes $p(\ell_i \mid G_{t_i})$

We turn to section 2.4. A lexeme token depends on its tag: draw ℓ_i from G_{t_i}, so $p(\ell_i \mid G_{t_i}) = G_{t_i}(\ell_i)$.

6.7 Inflections $p(s_i \mid H_{t_i,\ell_i})$

An inflection slot depends on its tagged lexeme: we draw s_i from H_{t_i,ℓ_i}, so $p(s_i \mid H_{t_i,\ell_i}) = H_{t_i,\ell_i}(s_i)$.

6.8 Spell-out $p(w_i \mid \pi_{t_i,\ell_i}(s_i))$

Finally, we generate the word w_i through a deterministic *spell-out* step.[11] Given the tag, lexeme, and inflection at position i, we generate the word w_i simply by looking up its spelling in the appropriate paradigm. So $p(w_i \mid \pi_{t_i,\ell_i}(s_i))$ is 1 if $w_i = \pi_{t_i,\ell_i}(s_i)$, else 0.

6.9 Collapsing the Assignment

Again, a *full* assignment's probability is the product of all the above factors (see drawing in Appendix B).

[11] To account for typographical errors in the corpus, the spell-out process could easily be made nondeterministic, with the observed word w_i derived from the correct spelling $\pi_{t_i,\ell_i}(s_i)$ by a noisy channel model (e.g., (Toutanova and Moore, 2002)) represented as a WFST. This would make it possible to analyze `brkoen` as a misspelling of a common or contextually likely word, rather than treating it as an unpronounceable, irregularly inflected neologism, which is presumably less likely.

But computationally, our sampler's state leaves the G_t *unspecified*. So its probability is the integral of $p(\text{assignment})$ over all possible G_t. As G_t appears only in the factors from headings 6.3 and 6.6, we can just integrate it out of *their* product, to get a collapsed sub-model that generates $p(\vec{\ell} \mid \vec{t}, \vec{\alpha})$ directly:

$$\int_{G_{\text{ADJ}}} \cdots \int_{G_{\text{VERB}}} d\mathbf{G} \left(\prod_{t \in \mathcal{T}} p(G_t \mid \alpha_t) \right) \left(\prod_{i=1}^{n} p(\ell_i \mid G_{t_i}) \right)$$
$$= p(\vec{\ell} \mid \vec{t}, \vec{\alpha}) = \prod_{i=1}^{n} p(\ell_i \mid \ell_1, \ldots \ell_{i-1}\ \vec{t}, \vec{\alpha})$$

where it turns out that the factor that generates ℓ_i is proportional to $|\{j < i : \ell_j = \ell_i \text{ and } t_j = t_i\}|$ if that integer is positive, else proportional to $\alpha_{t_i} G(\ell_i)$.

Metaphorically, each tag t is a *Chinese restaurant* whose *tables* are labeled with lexemes. The tokens are hungry *customers*. Each customer $i = 1, 2, \ldots, n$ enters restaurant t_i in turn, and ℓ_i denotes the label of the table she joins. She picks an occupied table with probability proportional to the number of previous customers already there, or with probability proportional to α_{t_i} she starts a new table whose label is drawn from G (it is novel with probability 1, since G gives infinitesimal probability to each old label).

Similarly, we integrate out the infinitely many lexeme-specific distributions $H_{t,\ell}$ from the product of 6.4 and 6.7, replacing it by the collapsed distribution

$$p(\vec{s} \mid \vec{\ell}, \vec{t}, \overrightarrow{\phi_t}, \overrightarrow{\alpha'}) \qquad \textit{[recall that } \overrightarrow{\phi_t} \textit{ determines } H_t\textit{]}$$
$$= \prod_{i=1}^{n} p(s_i \mid s_1, \ldots s_{i-1},\ \vec{\ell}, \vec{t}, \overrightarrow{\phi_t}, \overrightarrow{\alpha'})$$

where the factor for s_i is proportional to $|\{j < i : s_j = s_i \text{ and } (t_j, \ell_j) = (t_i, \ell_i)\}| + \alpha'_{t_i} H_{t_i}(s_i)$.

Metaphorically, each table ℓ in Chinese restaurant t has a fixed, finite set of *seats* corresponding to the inflections $s \in \mathcal{S}_t$. Each seat is really a bench that can hold any number of customers (tokens). When customer i chooses to sit at table ℓ_i, she also chooses a seat s_i at that table (see Fig. 2), choosing either an already occupied seat with probability proportional to the number of customers already in that seat, or else a random seat (sampled from H_{t_i} and not necessarily empty) with probability proportional to α'_{t_i}.

7 Inference and Learning

As section 3 explained, the learner alternates between a Monte Carlo E step that uses Gibbs sampling to

sample from the posterior of $(\vec{s}, \vec{\ell}, \vec{t})$ given $\vec{w}$ and the grammar variables, and an M step that adjusts the grammar variables to maximize the probability of the $(\vec{w}, \vec{s}, \vec{\ell}, \vec{t})$ samples given those variables.

7.1 Block Gibbs Sampling

As in Gibbs sampling for the DPMM, our sampler's basic move is to reanalyze token i (see section 3). This corresponds to making customer i invisible and then guessing where she is probably sitting—which restaurant t, table ℓ, and seat s?—given knowledge of w_i and the locations of all other customers.[12]

Concretely, the sampler guesses location (t_i, ℓ_i, s_i) with probability *proportional* to the product of

- $p(t_i \mid t_{i-1}, t_{i+1}, \vec{\tau})$ (from section 6.5)
- the probability (from section 6.9) that a new customer in restaurant t_i chooses table ℓ_i, given the *other* customers in that restaurant (and α_{t_i})[13]
- the probability (from section 6.9) that a new customer at table ℓ_i chooses seat s_i, given the *other* customers at that table (and $\overrightarrow{\phi_{t_i}}$ and α'_{t_i})[13]
- the probability (from section 3.3's belief propagation) that $\pi_{t_i,\ell_i}(s_i) = w_i$ (given $\vec{\theta}$).

We sample only from the (t_i, ℓ_i, s_i) candidates for which the last factor is non-negligible. These are found with the hash tables and FSAs of section 3.3.

7.2 Semi-Supervised Sampling

Our experiments also consider the semi-supervised case where a few **seed paradigms**—*type* data—are fully or partially observed. Our samples should also be conditioned on these observations. We assume that our supervised list of observed paradigms was generated by sampling from G_t.[14] We can modify our setup for this case: certain tables have a **host** who dictates the spelling of some seats and attracts appropriate customers to the table. See Appendix C.

7.3 Parameter Gradients

Appendix D gives formulas for the M step gradients.

[12] Actually, to improve mixing time, we choose a currently active lexeme ℓ uniformly at random, make *all* customers $\{i : \ell_i = \ell\}$ invisible, and sequentially guess where they are sitting.

[13] This is simple to find thanks to the exchangeability of the CRP, which lets us pretend that i entered the restaurant last.

[14] Implying that they are assigned to lexemes with non-negligible probability. We would learn nothing from a list of merely *possible* paradigms, since $\mathcal{L}_t$ is infinite and every conceivable paradigm is assigned to *some* $\ell \in \mathcal{L}_t$ (in fact many!).

	50 seed paradigms			100 seed paradigms		
Corpus size	0	10^6	10^7	0	10^6	10^7
Accuracy	89.9	90.6	**90.9**	91.5	92.0	**92.2**
Edit dist.	0.20	0.19	**0.18**	0.18	**0.17**	**0.17**

Table 2: Whole-word accuracy and edit distance of predicted inflection forms given the lemma. Edit distance to the correct form is measured in characters. Best numbers per set of seed paradigms in bold (statistically significant on our large test set under a paired permutation test, $p < 0.05$). Appendix E breaks down these results per inflection and gives an error analysis and other statistics.

8 Experiments

8.1 Experimental Design

We evaluated how well our model learns German verbal morphology. As *corpus* we used the first 1 million or 10 million words from WaCky (Baroni et al., 2009). For *seed and test paradigms* we used verbal inflectional paradigms from the CELEX morphological database (Baayen et al., 1995). We fully observed the seed paradigms. For each test paradigm, we observed the lemma type (Appendix C) and evaluated how well the system completed the other 21 forms (see Appendix E.2) in the paradigm.

We simplified inference by fixing the POS tag sequence to the automatic tags delivered with the WaCky corpus. The result that we evaluated for each variable was the value whose probability, averaged over the entire Monte Carlo EM run,[15] was highest. For more details, see (Dreyer, 2011).

All results are averaged over 10 different training/test splits of the CELEX data. Each split sampled 100 paradigms as seed data and used the remaining 5,415 paradigms for evaluation.[16] From the 100 paradigms, we also sampled 50 to obtain results with smaller seed data.[17]

8.2 Results

Type-based Evaluation. Table 2 shows the results of predicting verb inflections, when running with no corpus, versus with an unannotated corpus of size 10^6 and 10^7 words. Just using 50 seed paradigms, but

[15] This includes samples from before $\vec{\theta}$ has converged, somewhat like the voted perceptron (Freund and Schapire, 1999).

[16] 100 further paradigms were held out for future use.

[17] Since these seed paradigms are sampled uniformly from a set of CELEX paradigms, most of them are regular. We actually only used 90 and 40 for training, reserving 10 as development data for sanity checks and for deciding when to stop.

Bin	Frequency	# Verb Forms
1	0–9	116,776
2	10–99	4,623
3	100–999	1,048
4	1,000–9,999	95
5	10,000–	10
all	*any*	122,552

Table 3: The inflected verb forms from 5,615 inflectional paradigms, split into 5 token frequency bins. The frequencies are based on the 10-million word corpus.

Bin	50 seed paradigms			100 seed paradigms		
	0	10^6	10^7	0	10^6	10^7
1	90.5	91.0	**91.3**	92.1	92.4	**92.6**
2	78.1	**84.5**	84.4	80.2	**85.5**	85.1
3	71.6	**79.3**	78.1	73.3	**80.2**	79.1
4	57.4	61.4	**61.8**	57.4	**62.0**	59.9
5	20.7	**25.0**	**25.0**	20.7	**25.0**	**25.0**
all	52.6	57.5	**57.8**	53.4	**58.5**	57.8
all (e.d.)	1.18	1.07	**1.03**	1.16	1.02	**1.01**

Table 4: Token-based analysis: Whole-word accuracy results split into different frequency bins. In the last two rows, all predictions are included, weighted by the frequency of the form to predict. Last row is edit distance.

no corpus, gives an accuracy of 89.9%. By adding a corpus of 10 million words we reduce the error rate by 10%, corresponding to a one-point increase in absolute accuracy to 90.9%. A similar trend can be seen when we use more seed paradigms. Simply training on 100 seed paradigms, but not using a corpus, results in an accuracy of 91.5%. Adding a corpus of 10 million words to these 100 paradigms reduces the error rate by 8.3%, increasing the absolute accuracy to 92.2%. Compared to the large corpus, the smaller corpus of 1 million words goes more than half the way; it results in error reductions of 6.9% (50 seed paradigms) and 5.8% (100 seed paradigms). Larger unsupervised corpora should help by increasing coverage even more, although Zipf's law implies a diminishing rate of return.[18]

We also tested a baseline that simply inflects each morphological form according to the basic regular German inflection pattern; this reaches an accuracy of only 84.5%.

Token-based Evaluation. We now split our results into different bins: how well do we predict the spellings of frequently expressed (lexeme, inflection) pairs as opposed to rare ones? For example, the third person singular indicative of *give* (geben) is used significantly more often than the second person plural subjunctive of *bask* (aalen);[19] they are in different frequency bins (Table 3). The more frequent a form is in text, the more likely it is to be irregular (Jurafsky et al., 2000, p. 49).

The results in Table 4 show: Adding a corpus of either 1 or 10 million words increases our prediction accuracy across *all* frequency bins, often dramatically. All methods do best on the huge number of rare forms (Bin 1), which are mostly regular, and worst on on the 10 most frequent forms of the language (Bin 5). However, adding a corpus helps most in fixing the errors in bins with more frequent and hence more irregular verbs: in Bins 2–5 we observe improvements of up to almost 8% absolute percentage points. In Bin 1, the no-corpus baseline is already relatively strong.

Surprisingly, while we always observe gains from using a corpus, the gains from the 10-million-word corpus are sometimes smaller than the gains from the 1-million-word corpus, except in edit distance. Why? The larger corpus mostly adds new infrequent types, biasing $\vec{\theta}$ toward regular morphology at the expense of irregular types. A solution might be to model irregular classes with separate parameters, using the latent conjugation-class model of Dreyer et al. (2008).

Note that, by using a corpus, we even improve our prediction accuracy for forms and spellings that are *not* found in the corpus, i.e., *novel* words. This is thanks to improved grammar parameters. In the token-based analysis above we have already seen that prediction accuracy increases for rare forms (Bin 1). We add two more analyses that more explicitly show our performance on novel words. (a) We find all paradigms that consist of novel spellings only, i.e. none of the correct spellings can be found in the corpus.[20] The whole-word prediction accuracies for the models that use corpus size 0, 1 million, and 10 million words are, respectively, 94.0%, 94.2%, 94.4% using 50 seed paradigms, and 95.1%, 95.3%, 95.2% using 100 seed paradigms. (b) Another, sim-

[18] Considering the 63,778 distinct spellings from all of our 5,615 CELEX paradigms, we find that the smaller corpus contains 7,376 spellings and the 10× larger corpus contains 13,572.

[19] See Appendix F for how this was estimated from text.

[20] This is measured on the largest corpus used in inference, the 10-million-word corpus, so that we can evaluate all models on the same set of paradigms.

pler measure is the prediction accuracy on all forms whose correct spelling cannot be found in the 10-million-word corpus. Here we measure accuracies of 91.6%, 91.8% and 91.8%, respectively, using 50 seed paradigms. With 100 seed paradigms, we have 93.0%, 93.4% and 93.1%. The accuracies for the models that use a corpus are higher, but do not always steadily increase as we increase the corpus size.

The token-based analysis we have conducted here shows the strength of the corpus-based approach presented in this paper. While the integrated graphical models over strings (Dreyer and Eisner, 2009) can learn some basic morphology from the seed paradigms, the added corpus plays an important role in correcting its mistakes, especially for the more frequent, irregular verb forms. For examples of specific errors that the models make, see Appendix E.3.

9 Related Work

Our word-and-paradigm model seamlessly handles nonconcatenative and concatenative morphology alike, whereas most previous work in morphological knowledge discovery has modeled concatenative morphology only, assuming that the orthographic form of a word can be split neatly into stem and affixes—a simplifying asssumption that is convenient but often not entirely appropriate (Kay, 1987) (how should one segment English `stopping`, `hoping`, or `knives`?).

In **concatenative** work, Harris (1955) finds morpheme boundaries and segments words accordingly, an approach that was later refined by Hafer and Weiss (1974), Déjean (1998), and many others. The unsupervised segmentation task is tackled in the annual Morpho Challenge (Kurimo et al., 2010), where ParaMor (Monson et al., 2007) and Morfessor (Creutz and Lagus, 2005) are influential contenders. The Bayesian methods that Goldwater et al. (2006b, et seq.) use to segment between words might also be applied to segment within words, but have no notion of paradigms. Goldsmith (2001) finds what he calls *signatures*—sets of affixes that are used with a given set of stems, for example (*NULL*, `-er`, `-ing`, `-s`). Chan (2006) learns sets of morphologically related words; he calls these sets *paradigms* but notes that they are not substructured entities, in contrast to the paradigms we model in this paper. His models are restricted to concatenative and regular morphology.

Morphology discovery approaches that handle **nonconcatenative** and irregular phenomena are more closely related to our work; they are rarer. Yarowsky and Wicentowski (2000) identify inflection-root pairs in large corpora without supervision. Using similarity as well as distributional clues, they identify even irregular pairs like `take`/`took`. Schone and Jurafsky (2001) and Baroni et al. (2002) extract whole conflation sets, like "`abuse`, `abused`, `abuses`, `abusive`, `abusively`, ...," which may also be irregular. We advance this work by not only extracting pairs or sets of related observed words, but whole structured inflectional paradigms, in which we can also predict forms that have never been observed. On the other hand, our present model does not yet use contextual information; we regard this as a future opportunity (see Appendix G). Naradowsky and Goldwater (2009) add simple spelling rules to the Bayesian model of (Goldwater et al., 2006a), enabling it to handle some systematically nonconcatenative cases. Our finite-state transducers can handle more diverse morphological phenomena.

10 Conclusions and Future Work

We have formulated a principled framework for simultaneously obtaining morphological annotation, an unbounded morphological lexicon that fills complete structured morphological paradigms with observed and predicted words, and parameters of a nonconcatenative generative morphology model.

We ran our sampler over a large corpus (10 million words), inferring everything jointly and reducing the prediction error for morphological inflections by up to 10%. We observed that adding a corpus increases the absolute prediction accuracy on frequently occurring morphological forms by up to almost 8%. Future extensions to the model could leverage token context for further improvements (Appendix G).

We believe that a major goal of our field should be to build full-scale explanatory probabilistic models of language. While we focus here on inflectional morphology and evaluate the results in isolation, we regard the present work as a significant step toward a larger generative model under which Bayesian inference would reconstruct other relationships as well (e.g., inflectional, derivational, and evolutionary) among the words in a family of languages.

References

A. C. Albright. 2002. *The Identification of Bases in Morphological Paradigms*. Ph.D. thesis, University of California, Los Angeles.

D. Aldous. 1985. Exchangeability and related topics. *École d'été de probabilités de Saint-Flour XIII*, pages 1–198.

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2(6):1152–1174.

R. H Baayen, R. Piepenbrock, and L. Gulikers. 1995. The CELEX lexical database (release 2)[cd-rom]. *Philadelphia, PA: Linguistic Data Consortium, University of Pennsylvania [Distributor]*.

M. Baroni, J. Matiasek, and H. Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proc. of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 48–57.

M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. 2009. The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

David Blackwell and James B. MacQueen. 1973. Ferguson distributions via Pòlya urn schemes. *The Annals of Statistics*, 1(2):353–355, March.

David M. Blei and Peter I. Frazier. 2010. Distance-dependent Chinese restaurant processes. In *Proc. of ICML*, pages 87–94.

E. Chan. 2006. Learning probabilistic paradigms for morphology in a latent class model. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology at HLT-NAACL*, pages 69–78.

M. Creutz and K. Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. *Computer and Information Science, Report A*, 81.

H. Déjean. 1998. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proc. of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*, pages 295–298.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *Proc. of EMNLP*, Singapore, August.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*, Honolulu, Hawaii, October.

Markus Dreyer. 2011. *A Non-Parametric Model for the Discovery of Inflectional Paradigms from Plain Text Using Graphical Models over Strings*. Ph.D. thesis, Johns Hopkins University.

T.S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, 1(2):209–230.

Y. Freund and R. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

S. Goldwater, T. Griffiths, and M. Johnson. 2006a. Interpolating between types and tokens by estimating power-law generators. In *Proc. of NIPS*, volume 18, pages 459–466.

S. Goldwater, T. L. Griffiths, and M. Johnson. 2006b. Contextual dependencies in unsupervised word segmentation. In *Proc. of COLING-ACL*.

P.J. Green. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711.

M. A Hafer and S. F Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10:371–385.

Z. S. Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

G.E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Klara Janecki. 2000. *300 Polish Verbs*. Barron's Educational Series.

E. T. Jaynes. 2003. *Probability Theory: The Logic of Science*. Cambridge Univ Press. Edited by Larry Bretthorst.

D. Jurafsky, J. H. Martin, A. Kehler, K. Vander Linden, and N. Ward. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. MIT Press.

M. Kay. 1987. Nonconcatenative finite-state morphology. In *Proc. of EACL*, pages 2–10.

M. Kurimo, S. Virpioja, V. Turunen, and K. Lagus. 2010. Morpho Challenge competition 2005–2010: Evaluations and results. In *Proc. of ACL SIGMORPHON*, pages 87–95.

P. H. Matthews. 1972. *Inflectional Morphology: A Theoretical Study Based on Aspects of Latin Verb Conjugation*. Cambridge University Press.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. 2007. ParaMor: Minimally supervised induction of paradigm structure and morphological analysis. In *Proc. of ACL SIGMORPHON*, pages 117–125, June.

J. Naradowsky and S. Goldwater. 2009. Improving morphology induction by learning spelling rules. In *Proc. of IJCAI*, pages 1531–1536.

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900.

P. Schone and D. Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proc. of NAACL*, volume 183, pages 183–191.

J. Sethuraman. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4(2):639–650.

N. A. Smith, D. A. Smith, and R. W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of HLT-EMNLP*, pages 475–482, October.

G. T. Stump. 2001. *Inflectional Morphology: A Theory of Paradigm Structure*. Cambridge University Press.

Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. of ACL*.

K. Toutanova and R.C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proc. of ACL*, pages 144–151.

D. Yarowsky and R. Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proc. of ACL*, pages 207–216, October.

Multilayer Sequence Labeling

Ai Azuma **Yuji Matsumoto**
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
{ai-a,matsu}@is.naist.jp

Abstract

In this paper, we describe a novel approach to cascaded learning and inference on sequences. We propose a weakly joint learning model on cascaded inference on sequences, called multilayer sequence labeling. In this model, inference on sequences is modeled as cascaded decision. However, the decision on a sequence labeling sequel to other decisions utilizes the features on the preceding results as marginalized by the probabilistic models on them. It is not novel itself, but our idea central to this paper is that the probabilistic models on succeeding labeling are viewed as indirectly depending on the probabilistic models on preceding analyses. We also propose two types of efficient dynamic programming which are required in the gradient-based optimization of an objective function. One of the dynamic programming algorithms resembles back propagation algorithm for multilayer feed-forward neural networks. The other is a generalized version of the forward-backward algorithm. We also report experiments of cascaded part-of-speech tagging and chunking of English sentences and show effectiveness of the proposed method.

1 Introduction

Machine learning approach is widely used to classify instances into discrete categories. In many tasks, however, some set of inter-related labels should be decided simultaneously. Such tasks are called structured prediction. Sequence labeling is the simplest subclass of structured prediction problems. In sequence labeling, the most likely one among all the possible label sequences is predicted for a given input. Although sequence labeling is the simplest subclass, a lot of real-world tasks are modeled as problems of this simplest subclass. In addition, it might offer valuable insight and a toehold for more general and complex structured prediction problems. Many models have been proposed for sequence labeling tasks, such as Hidden Markov Models (HMM), Conditional Random Fields (CRF) (Lafferty et al., 2001), Max-Margin Markov Networks (Taskar et al., 2003) and others. These models have been applied to lots of practical tasks in natural language processing (NLP), bioinformatics, speech recognition, and so on. And they have shown great success in recent years.

In real-world tasks, it is often needed to cascade multiple predictions. A cascade of predictions here means the situation in which some of predictions are made based upon the results of other predictions. Sequence labeling is not an exception. For example, in NLP, we perform named entity recognition or base-phrase chunking for given sentences based on part-of-speech (POS) labels predicted by another sequence labeler. Natural languages are especially interpreted to have a hierarchy of sequential structures on different levels of abstraction. Therefore, many tasks in NLP are modeled as a cascade of sequence predictions.

If a prediction is based upon the result of another prediction, we call the former upper stage and the latter lower stage.

Methods pursued for a cascade of predictions – including sequence predictions, of course–, are desired to perform certain types of capability. One de-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 628–637,
Edinburgh, Scotland, UK, July 27–31, 2011.

sired capability is rich forward information propagation, that is, the learning and estimation on each stage of predictions should utilize rich information of the results of lower stages whenever possible. "Rich information" here includes next bests and confidence information of the results of lower stages. Another is backward information propagation, that is, the rich annotated data on an upper stage should affect the models on lower stages retroactively.

Many current systems for a cascade of sequence predictions adopt a simple 1-best feed-forward approach. They simply take the most likely output at each prediction stage and transfer it to the next upper stage. Such a framework can maximize reusability of existing sequence labeling systems. On the other hand, it exhibits a strong tendency to propagate errors to upper labelers.

Typical improvement on the 1-best approach is to keep k-best results in the cascade of predictions. However, the larger k becomes, the more difficult it is to enumerate and maintain the k-best results. It is particularly prominent in sequence labeling.

The essence of this orientation is that the labeler on an upper stage utilizes the information of all the possible output candidates on lower stages. However, the size of the output space can become quite large in sequence labeling. It effectively forbids explicit enumeration of all possible outputs, so it is required to represent all the labeling possibilities compactly or employ some approximation schemes. Several studies are in this direction. In the method proposed in Finkel et al. (2006), a cascades of sequence predictions is viewed as a Bayesian network, and sample sequences are drawn at each stage according to the output distribution. The samples are then used to estimate the entire distribution of the cascade. In the method proposed in Bunescu (2008), an upper labeler uses the probabilities marginalized on the parts of the output sequences on lower stages as weights for the features. The weighted features are integrated in the model of the labeler on the upper stage. A k-best approach (e.g., (Collins and Duffy, 2002)) and the methods mentioned above are effective to improve the forward information propagation. However, they can never contribute on backward information propagation.

To improve the both directions of information propagation, Some studies propose the joint learning of multiple sequence labelers. Sutton et al. (2007) proposes the joint learning method in case where multiple labels are assigned to each time slice of the input sequences. It enables simultaneous learning and estimation of multiple sequence labelings on the same input sequences, where time slices of the outputs of all the out sequences are regularly aligned. However, it puts the distribution of states into Bayesian networks with cyclic dependencies, and exact inference is not tractable in such a model in general. Therefore, it requires some approximate inference algorithms in learning or predictions. Moreover, it only considers the cases where labels of an input sequence and all output sequences are regularly aligned. It is not clear how to build a joint labeling model which handles irregular output label sequences like semi-Markov models (Sarawagi and Cohen, 2005).

In this paper, we propose a middle ground for a cascade of sequence predictions. The proposed method adopts the basic idea of Bunescu (2008). We first assume that the model on all the sequence labeling stages is probabilistic one. In modeling of an upper stage, a feature is weighted by the marginal probability of the fragment of the outputs from a lower stage. However, this is not novel itself because it is just a paraphrase of Bunescu's core idea. Our intuition behind the proposed method is as follows. Features integrated in the model on each stage are weighted by the marginal probabilities of the fragments of the outputs on lower stages. So, if the output distributions on lower stages change, the marginal probabilities of any fragments also change, and this in turn can change the value of the features on the upper stage. In other words, the features on an upper stage indirectly depend on the models on the lower stages. Based on this intuition, the learning procedure of the model on an upper stage can affect not only direct model parameters, but also the weights of the features by changing the model on the lower stages. Supervised learning based on annotated data on an upper stage may affect the model or model parameters on the lower stages. It could be said that the information of annotation data on an upper stage is propagated back to the model on lower stages.

In the next section, we describe the formal nota-

tion of our model. In Section 3, we propose an optimization procedure according to the intuition noted above. In Section 4, we report an experimental result of our method. The proposed method shows some improvements on a real-world task in comparison with ordinary methods.

2 Formalization

In this section, we introduce the formal notation of our model. Hereafter, for the sake of simplicity, we only describe the simplest case in which there are just two stages, one lower stage of sequence labeling named L_1 and one upper stage of sequence labeling named L_2. In L_1, the most likely one among a set of possible sequences is predicted for a given input $\mathbf{x}$. L_2 is also a sequence labeling stage for the same input $\mathbf{x}$ and the output of L_1. No assumption is made on the structure of $\mathbf{x}$. The information of $\mathbf{x}$ is totally encoded in feature functions. It is only assumed that the output spaces of both L_1 and L_2 are conditioned on the initial input $\mathbf{x}$.

First of all, we describe the formalization of the probabilistic model for L_1. The model for L_1 per se is the same as ordinary ones for sequence labeling. For a given input $\mathbf{x}$, consider a directed acyclic graph (DAG) $G_1 = (V_1, E_1)$. A source of a DAG G is a node whose in-degree is equal to zero. A sink of a DAG G is nodes whose out-degree is equal to zero. Let $\mathrm{src}(G)$, $\mathrm{snk}(G)$ denote the set of source and sink nodes in G, respectively. A successful path of a DAG G is defined as a directed path on G whose starting node is a source and end node is a sink. If $\mathbf{y}$ denotes a path on a DAG, let $\mathbf{y}$ also denote the set of all the arcs appearing on $\mathbf{y}$ for the sake of shorthand. We denote the set of all the possible successful paths on G_1 by $\mathbf{Y}_1$. The space of the output candidates for L_1 is exactly equal to $\mathbf{Y}_1$. For the modeling of L_1, it is assumed that features of the form $f_{\langle 1,k_1,e_1,\mathbf{x}\rangle} \in \mathbb{R}$ $(k_1 \in \mathcal{K}_1, e_1 \in E_1)$ are allowed to be used. Here, $\mathcal{K}_1$ is the index set of the feature types for L_1. Such a feature can capture an aspect of the correlation between adjacent nodes. We call this kind of features input features for L_1. This naming is used to distinguish them from another kind of features defined on L_1, which comes later. Although features on V_1 can be also defined, they are totally omitted in this paper for brevity. Hereafter, if a symbol has subscripts, then missing subscript indicates a set that range over the omitted subscript. For example, $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle} \stackrel{\mathrm{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{k_1\in\mathcal{K}_1}$, $\mathbf{f}_{\langle 1,k_1,\mathbf{x}\rangle} \stackrel{\mathrm{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{e_1\in E_1}$, $\mathbf{f}_{\langle 1,\mathbf{x}\rangle} \stackrel{\mathrm{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{k_1\in\mathcal{K}_1,e_1\in E_1}$, and so on. The probabilistic model on L_1 forms the log-linear model, that is,

$$P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \stackrel{\mathrm{def}}{\equiv} \frac{1}{Z_1(\mathbf{x};\boldsymbol{\theta}_1)} \exp\left(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right) \quad (\mathbf{y}_1 \in \mathbf{Y}_1) \ , \tag{1}$$

where $\theta_{\langle 1,k_1\rangle} \in \mathbb{R}$ $(k_1 \in \mathcal{K}_1)$ is the weight for the feature of the same index k_1, and the k_1-th element of $\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}$, $F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x}\rangle} \stackrel{\mathrm{def}}{\equiv} \sum_{e_1\in\mathbf{y}_1} f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}$. Dot operator $(\cdot)$ denotes the inner product with respect to the subscripts commonly missing in both operands. Z_1 is the partition function for P_1, defined as

$$Z_1(\mathbf{x};\boldsymbol{\theta}_1) \stackrel{\mathrm{def}}{\equiv} \sum_{\mathbf{y}_1\in\mathbf{Y}_1} \exp\left(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right) \ . \tag{2}$$

It is worth noting that this formalization subsumes both directed and undirected linear-chain graphical models, which are the most typical models for sequence labeling, including HMM and CRF. That is, if the elements of V_1 are aligned into regular time slices, and the nodes in each time slice are associated with possible assignments of labels for that time, we obtain the representation equivalent to the ordinary linear-chain graphical models, in which all possible label assignments for each state are expanded. In such configuration, all the possible successful paths defined in our notation have strict one-to-one correspondence to all the possible joint assignments of labels in linear-chain graphical models. We purposely employ this DAG-based notation because; it is convenient to describe the models and algorithms for our purpose, it allows for labels to stay in arbitrary time as in semi-Markov models, and it is easily extended to models for a set of trees instead of sequences by replacing the graph-based notation with hypergraph-based notation.

Next, we formalize the probabilistic model on the upper stage L_2. Like L_1, consider a DAG $G_2 = (V_2, E_2)$ conditioned on the input $\mathbf{x}$, and the set of all the possible successful paths on G_2, denoted $\mathbf{Y}_2$. The space of the output candidates for L_2 becomes $\mathbf{Y}_2$.

The form of the features available in designing the probabilistic model for L_2, denoted by P_2, is the key of this paper. A feature on an arc $e_2 \in E_2$ can access local characteristics of the confidence-rated superposition of the L_1's outputs, in addition to the information of the input $\mathbf{x}$. To formulate local characteristics of the superposition of the L_1's outputs, we first define output features of L_1, denoted by $h_{\langle 1,k'_1,e_1\rangle} \in \mathbb{R} \quad (k'_1 \in \mathcal{K}'_1,\, e_1 \in E_1)$. Here, $\mathcal{K}'_1$ is the index set of the output feature types of L_1. Before the output features are integrated into the model for L_2, they all are confidence-rated with respect to P_1, that is, each output feature $h_{\langle 1,k'_1,e_1\rangle}$ is numerically rated by the estimated probabilities summed over the sequences emitting that feature. More formally, all the L_1's output features are integrated in features for P_2 in the form of the marginalized output features, which are defined as follows;

$$\bar{h}_{\langle 1,k'_1,e_1\rangle}(\boldsymbol{\theta}_1) \stackrel{\text{def}}{\equiv} h_{\langle 1,k'_1,e_1\rangle} P_1(e_1|\mathbf{x};\boldsymbol{\theta}_1) \qquad \left(k'_1 \in \mathcal{K}'_1,\, e_1 \in E_1\right) \quad , \tag{3}$$

where

$$\begin{aligned} P_1(e_1|\mathbf{x};\boldsymbol{\theta}_1) &\stackrel{\text{def}}{\equiv} \sum_{\mathbf{y}_1 \sim e_1} P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \\ &= \sum_{\mathbf{y}_1 \in \mathbf{Y}_1} \delta_{e_1 \in \mathbf{y}_1} P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \qquad (e_1 \in E_1) \quad . \end{aligned} \tag{4}$$

Here, the notation $\sum_{\mathbf{y}_1 \sim e_1}$ represents the summation over sequences consistent with an arc $e_1 \in E_1$, that is, the summation over the set $\{\mathbf{y}_1 \in \mathbf{Y}_1 \mid e_1 \in \mathbf{y}_1\}$. $\delta_{\mathcal{P}}$ denotes the indicator function for a predicate $\mathcal{P}$. The input features for P_2 on an arc $e_2 \in E_2$ are permitted to arbitrarily combine the information of $\mathbf{x}$ and the L_1's marginalized output features $\bar{\mathbf{h}}_1$, in addition to the local characteristics of the arc at hand e_2. In summary, an input feature for L_2 on an arc $e_2 \in E_2$ is of the form

$$f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big) \in \mathbb{R} \qquad (k_2 \in \mathcal{K}_2) \quad , \tag{5}$$

where $\mathcal{K}_2$ is the index set of the input feature types for L_2. To make the optimization procedure feasible, smoothness condition on any L_2's input feature is assumed with respect to all the L_1's output features, that is, $\frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial h_{\langle 1,k'_1,e_1\rangle}}$ is always guaranteed to exist for $^{\forall}k'_1,\, e_1,\, k_2,\, e_2$. For example, additions and multiplications between some elements of $\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)$ can appear in the definition of L_2's input features. For given input features $\mathbf{f}_{\langle 2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big)$ and parameters $\theta_{\langle 2,k_2\rangle} \in \mathbb{R} \quad (k_2 \in \mathcal{K}_2)$, the probabilistic model for L_2 is defined as follows;

$$\begin{aligned} &P_2(\mathbf{y}_2|\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) \\ &\quad \stackrel{\text{def}}{\equiv} \frac{1}{Z_2(\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)} \exp\big(\boldsymbol{\theta}_2 \cdot \mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big)\big) \\ &\qquad\qquad (\mathbf{y}_2 \in \mathbf{Y}_2) \quad , \end{aligned} \tag{6}$$

where $F_{\langle 2,k_2,\mathbf{y}_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big) \stackrel{\text{def}}{\equiv} \sum_{e_2 \in \mathbf{y}_2} f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big)$ and Z_2 is the partition function of P_2, defined by

$$\begin{aligned} &Z_2(\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) \\ &\quad \stackrel{\text{def}}{\equiv} \sum_{\mathbf{y}_2 \in \mathbf{Y}_2} \exp\big(\boldsymbol{\theta}_2 \cdot \mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big)\big) \quad . \end{aligned} \tag{7}$$

The definition of P_2 (6) reveals one of the most important points in this paper. P_2 is viewed not only as the function of the ordinary direct parameters $\boldsymbol{\theta}_2$ but also as the function of $\boldsymbol{\theta}_1$, which represents the parameters for the L_1's model, through the intermediate variables $\bar{\mathbf{h}}_1$. So optimization procedure on P_2 may affect the determination of the values not only of the direct parameters $\boldsymbol{\theta}_2$ but also of the indirect ones $\boldsymbol{\theta}_1$.

If the result of L_1 is reduced to the single golden output $\tilde{\mathbf{y}}_1$, i.e. $P_1(\mathbf{y}_1|\mathbf{x}) = \delta_{\mathbf{y}_1 = \tilde{\mathbf{y}}_1}$, the definitions above boil down to the formulation of the simple 1-best feed forward architecture.

3 Optimization Algorithm

In this section, we describe optimization procedure for the model formulated in the previous section. Let $\mathcal{D} = \{\langle \hat{\mathbf{x}}, \langle G_1, \hat{\mathbf{y}}_1\rangle, \langle G_2, \hat{\mathbf{y}}_2\rangle\rangle_m\}_{m=1,2,\cdots,M}$ denote annotated data for the supervised learning of the model. Here, $\langle G_1, \hat{\mathbf{y}}_1\rangle$ is a pair of a DAG and correctly annotated successful sequence for L_1. The same holds for $\langle G_2, \hat{\mathbf{y}}_2\rangle$. For given $\mathcal{D}$, we can define the conditional log-likelihood function on L_1 and L_2 respectively, that is,

$$\begin{aligned} &\mathcal{L}_1(\boldsymbol{\theta}_1;\mathcal{D}) \\ &\quad \stackrel{\text{def}}{\equiv} \sum_{\langle \hat{\mathbf{x}}, \hat{\mathbf{y}}_1\rangle \in \mathcal{D}} \log\left(P_1(\hat{\mathbf{y}}_1|\hat{\mathbf{x}};\boldsymbol{\theta}_1)\right) - \frac{|\boldsymbol{\theta}_1|}{2{\sigma_1}^2} \end{aligned} \tag{8}$$

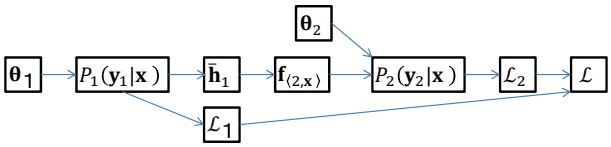


Figure 1: Computation Graph of the Proposed Model

and

$$\mathcal{L}_2\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; \mathcal{D}\right) \stackrel{\text{def}}{\equiv} \sum_{\langle \hat{\mathbf{x}}, \hat{\mathbf{y}}_2 \rangle \in \mathcal{D}} \log\left(P_2\left(\hat{\mathbf{y}}_2 | \hat{\mathbf{x}}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\right)\right) - \frac{|\boldsymbol{\theta}_2|}{2{\sigma_2}^2} \ . \tag{9}$$

Here, ${\sigma_1}^2$, ${\sigma_2}^2$ are the variances of the prior distributions of the parameters. For the sake of simplicity, we set the prior distribution as the zero-mean univariance Gaussian. To optimize the both probabilistic models P_1 and P_2 jointly, we also define the joint conditional log-likelihood function

$$\mathcal{L}\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; \mathcal{D}\right) \stackrel{\text{def}}{\equiv} \mathcal{L}_1 + \mathcal{L}_2 \ . \tag{10}$$

The parameter values to be learned are the ones that (possibly locally) maximize this objective function. Note that this objective function is not guaranteed to be globally convex.

We employ gradient-based parameter optimization here. Optimization procedure repeatedly searches a direction in the parameter space which is ascendent with respect to the objective function, and updates the parameter values into that direction by small advances. Many existing optimization routines like steepest descent or conjugation gradient do that job only by giving the objective value and gradients on parameter values to be updated. So, the optimization problem here boils down to the calculation of the objective value and gradients on given parameter values.

Before entering the detailed description of the algorithm for calculating the objective function and gradients, we note the functional relations among the objective function and previously defined variables. The diagram shown in Figure 1 illustrates the functional relations among the parameters, input and output feature functions, models, and objective function. The variables at the head of a directed arrow in the figure is directly defined in terms of the ones at the tail of the same arrow. The value of the objective function on given parameter values can be calculated in order of the arrows shown in the diagram. On the other hand, the parameter gradients are calculated step-by-step in reverse order of the arrows. The functional relations illustrated in the Figure 1 ensure some forms of the chain rule of differentiation among the variables. The chain rule is iteratively used to decompose the calculation of the gradients into a divide-and-conquer fashion. These two directions of stepwise computation are analogous to the forward and back propagation for multilayer feedforward neural networks, respectively.

Algorithm 1 shows the whole picture of the gradient-based optimization procedure for our model. We first describe the flow to calculate the objective value for a given parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, which is shown from line 2 through 4 in Algorithm 1. The values of marginalized output features $\bar{\mathbf{h}}_{\langle 1,\mathbf{x}\rangle}$ can be calculated by (3). Because they are the simple marginals of features, the ordinary forward-backward algorithm (hereafter, abbreviated as "F-B") on G_1 offers an efficient way to calculate their values. Although nothing definite about the forms of the input features for L_2 is presented in this paper, $\mathbf{f}_{\langle 2,\mathbf{x}\rangle}$ can be calculated once the values of $\bar{\mathbf{h}}_{\langle 1,\mathbf{x}\rangle}$ have been obtained. Finally, $\mathcal{L}_1$, $\mathcal{L}_2$ and then $\mathcal{L}$ are easy to calculate because they are no different from the ordinary log-likelihood computation.

Now we describe the algorithm to calculate the parameter gradients,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_1} = \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\theta}_1} + \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1}, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_2} = \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_2} \ . \tag{11}$$

Line 5 through line 7 in Algorithm 1 describe the gradient computation. The terms $\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\theta}_1}$ and $\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_2}$ in (11) become the same forms that appear in the ordinary CRF optimization, i.e., the difference between the empirical frequencies of the features and the model expectations of them,

$$\begin{aligned} \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\theta}_1} &= \tilde{E}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] - E_{P_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] - \frac{|\boldsymbol{\theta}_1|}{{\sigma_1}^2} \ , \\ \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_2} &= \tilde{E}\left[\mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\right] - E_{P_2}\left[\mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\right] - \frac{|\boldsymbol{\theta}_2|}{{\sigma_2}^2} \ . \end{aligned} \tag{12}$$

These calculations are performed by the ordinary F-B on G_1 and G_2, respectively. Using the chain rule of differentiation derived from the functional relations illustrated in Figure 1, the remaining term $\frac{\partial L_2}{\partial \boldsymbol{\theta}_1}$

Algorithm 1 Gradient-based optimization of the model parameters

Input: $\boldsymbol{\theta}_1,\, \boldsymbol{\theta}_2$

Output: $\arg\max_{\langle \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \rangle} \mathcal{L}$

1: **while** $\boldsymbol{\theta}_1$ or $\boldsymbol{\theta}_2$ changes significantly **do**
2: calculate Z_1 by (2), $\bar{\mathbf{h}}_1$ by (3) with the F-B on G_1, and then $\mathcal{L}_1$ by (8)
3: calculate $\mathbf{f}_{\langle 2,\mathbf{x} \rangle}$ according to their definitions
4: calculate Z_2 by (7) with the F-B on G_2, and then $\mathcal{L}_2$ by (9) and $\mathcal{L}$ by (10)
5: calculate $\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\theta}_1}$ and $\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_2}$ by (12) with the F-B on G_1 and G_2, respectively
6: calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{f}_{\langle 1,\mathbf{x} \rangle}}$ by (16) with the F-B on G_2, $\frac{\partial \mathbf{f}_{\langle 1,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1}$, and them $\frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_1} = \frac{\partial \mathcal{L}}{\partial \mathbf{f}_{\langle 1,\mathbf{x} \rangle}} \cdot \frac{\partial \mathbf{f}_{\langle 1,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1}$
7: calculate $\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1}$ by (18) with Algorithm 2
8: $\langle \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \rangle \leftarrow$ **update-parameters**$\Big(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathcal{L}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_1}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_2}\Big)$
9: **end while**

in (11) can be decomposed as follows;

$$\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1} = \frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}} \cdot \frac{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}}{\partial \boldsymbol{\theta}_1} = \frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}} \cdot \frac{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1} \cdot \frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1} \ . \tag{13}$$

Note that Leibniz's notation here denotes a Jacobian with the index sets omitted in the numerator and the denominator, for example,

$$\frac{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1} \stackrel{\text{def}}{=} \left\{ \frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x} \rangle}}{\partial h_{\langle 1,k'_1,e_1 \rangle}} \right\}_{k_2 \in \mathcal{K}_2, e_2 \in E_2, k'_1 \in \mathcal{K}'_1, e_1 \in E_1} \tag{14}$$

And also recall that dot operators here stand for the inner product with respect to the index sets commonly omitted in both operands, for example,

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_2} \cdot \frac{\partial \mathbf{f}_2}{\partial \bar{\mathbf{h}}_1} = \sum_{k_2 \in \mathcal{K}_2, e_2 \in E_2} \frac{\partial \mathcal{L}_2}{\partial f_{\langle 2,k_2,e_2,\mathbf{x} \rangle}} \cdot \frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1} \ . \tag{15}$$

We describe the manipulation of each factor in the right side of (13) in turn. Noting $\frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x} \rangle}}{\partial f_{\langle 2,\grave{k}_2,\grave{e}_2,\mathbf{x} \rangle}} = \delta_{k_2=\grave{k}_2} \delta_{e_2=\grave{e}_2}$, each element of the first factor of (13) $\frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}}$ can be transformed as follows;

$$\frac{\partial \mathcal{L}_2}{\partial f_{\langle 2,k_2,e_2,\mathbf{x} \rangle}} = \theta_{\langle 2,k_2 \rangle} \sum_{\langle \hat{\mathbf{x}}, \hat{\mathbf{y}}_2 \rangle \in \mathcal{D}} \big(\delta_{e_2 \in \hat{\mathbf{y}}_2} - P_2(e_2 | \hat{\mathbf{x}}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \big) \ . \tag{16}$$

$P_2(e_2|\hat{\mathbf{x}}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, the marginal probability on e_2, can be obtained as a by-product of the F-B for (12).

As described in the previous section, it is assumed that the values of the second factor $\frac{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1}$ is guaranteed to exists for any given $\boldsymbol{\theta}_1$, and the procedure for calculating them is fixed in advance. The procedure for some of concrete features is exemplified in the previous section.

From the definition of $\bar{\mathbf{h}}_1$ (3), each element of the third factor of (13) $\frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1}$ becomes

$$\frac{\partial \bar{h}_{\langle 1,k'_1,e_1 \rangle}}{\partial \theta_{\langle 1,k_1 \rangle}} = h_{\langle 1,k'_1,e_1 \rangle} \mathrm{Cov}_{P_1(\mathbf{y}_1|\mathbf{x})} \big[\delta_{e_1 \in \mathbf{y}_1},\, F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x} \rangle} \big] \ . \tag{17}$$

There exists efficient dynamic programming to calculate the covariance value (17) (without goint into that detail because it is very similar to the one shown later in this paper), and of course we can run such dynamic programming for $^\forall k'_1 \in \mathcal{K}'_1,\ e_1 \in E_1$. However, the size of the Jacobian $\frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1}$ is equal to $|\mathcal{K}'_1| \times |E_1| \times |\mathcal{K}_1|$. Since it is too large in many tasks likely to arise in practice, we should avoid to calculate all the elements of this Jacobian in a straightforward way. Instead of such naive computation, if the values of $\frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}}$ and $\frac{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1}$ are obtained, then we can compute $\frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_1} = \frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}} \cdot \frac{\partial \mathbf{f}_{\langle 2,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1}$, and from (13)

and (17),

$$\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1} = \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_1} \cdot \frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1} = E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle}\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] - E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle}\right] E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \quad , \tag{18}$$

where $H'_{\langle 1,\mathbf{y}_1\rangle} \stackrel{\text{def}}{\equiv} \sum_{e_1 \in \mathbf{y}_1} \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$. In other words, $\frac{\partial \mathcal{L}_2}{\partial \theta_{\langle 1,k_1\rangle}}$ becomes the covariance between the k_1-th input feature for L_1 and the hypothetical feature $h'_{\langle 1,e_1\rangle} \stackrel{\text{def}}{\equiv} \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$.

The final problem is to derive an efficient way to compute the first term of (18). The second term of (18) can be calculated by the ordinary F-B because it consists of the marginals of arc features. There are two derivations of the algorithm for calculating the first term. We describe briefly the both derivations.

One is a variant of the F-B on the expectation semi-ring proposed in Li and Eisner (2009). First, the F-B is generalized to the expectation semi-ring with respect to the hypothetical feature $h'_{\langle 1,e_1\rangle}$, and by summing up the marginals of the feature vectors $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}$ on all the arcs under the distribution of the semi-ring, then we obtain the expectation of the feature vector $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}$ on the semi-ring potential. This expectation is equal to the first term of (18). [1]

Another derivation is to apply the automatic differentiation (AD)(Wengert, 1964; Corliss et al., 2002) on the F-B calculating $E_{P_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]$. It exploits the fact that $\frac{\partial}{\partial \lambda} E_{P'_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]\Big|_{\lambda=0}$ is equal to the first term of (18), where $\lambda \in \mathbb{R}$ is a dummy parameter, and $P'_1(\mathbf{y}_1|\mathbf{x}) \stackrel{\text{def}}{\equiv} \frac{1}{Z_1}\exp\left(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle} + \lambda H'_{\langle 1,\mathbf{y}_1\rangle}\right)$. It is easy to derive the F-B for calculating the value $E_{P'_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]\Big|_{\lambda=0}$. AD transforms this F-B into another algorithm for calculating the differentiation w.r.t. λ evaluated at the point $\lambda = 0$. This transformation is achieved in an automatic manner, by replacing all appearances of λ in the F-B with a dual number $\lambda + \varepsilon$. The dual number is a variant of the complex number, with a kind of the imaginary unit ε with the property $\varepsilon^2 = 0$. Like the usual complex numbers, the arithmetic operations and the exponential function are generalized to the dual numbers, and the ordinary F-B is also generalized to the dual numbers. The imaginary part of the resulting values is equal to the needed differentiation. [2] Anyway, these two derivations lead to the same algorithm, and the resulting algorithm is shown as Algorithm 2.

The final line in the loop of Algorithm 1 can be implemented by various optimization routines and line search algorithms.

The time and space complexity to compute the objective and gradient values for given parameter vectors $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ is the same as that for that for Bunescu (2008), up to a constant factor. Because the calculation of the objective function is essentially the same as that for Bunescu (2008), and in gradient computation, the time complexity of Algorithm 1 is the same as that for the ordinary F-B (up to a constant factor), and the proposed optimization procedure is only required to store additional scalar values $h'_{\langle 1,e_1\rangle}$ on each G_1's arc.

4 Experiment

We examined effectiveness of the method proposed in this paper on a real task. The task is to annotate the POS tags and to perform base-phrase chunking on English sentences.

Base-phrase chunking is a task to classify continuous subsequences of words into syntactic categories. This task is performed by annotating a chunking label on each word (Ramshaw and Marcus, 1995). The types of chunking label consist of "Begin-*Category*", which represents the beginning of a chunk, "Inside-*Category*", which represents the inside of a chunk, and "Other." Usually, POS labeling runs first before base-phrase chunking is performed. Therefore, this task is a typical interesting case where a sequence labeling depends on the output from other sequence labelers.

The data used for our experiment consist of English sentences from the Penn Treebank project (Marcus et al., 1993) consisting of 10948 sentences and 259104 words. We divided them into two groups, training data consisting of 8936 sentences and 211727 words and test data consisting of 2012

[1] For the detailed description, see Li and Eisner (2009) and its references.

[2] For example, Berz (1992) gives a detailed description of the reason why the dual number is used for this purpose.

Algorithm 2 Forward-backward Algorithm for Calculating Feature Covariances

Input: $\mathbf{f}_{\langle 1,\mathbf{x}\rangle}$, $\phi_{e_1} \stackrel{\text{def}}{\equiv} \exp\left(\boldsymbol{\theta}_1 \cdot \mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}\right)$, $h'_{e_1} \stackrel{\text{def}}{\equiv} \frac{\partial \mathcal{L}_2}{\partial \mathbf{h}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$

Output: $q_{k_1} = \mathrm{Cov}_{P(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle},\, F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \qquad \left(^\forall k_1 \in \mathcal{K}_1\right)$

for $^\forall v_1 \in \mathrm{src}(G_1)$, $\alpha_{v_1} \leftarrow 1$, $\alpha'_{v_1} \leftarrow 1$
for all $v_1 \in V_1$ in a topological order **do**
  $\mathrm{prev} \leftarrow \{x \in V_1 \mid (x, v_1) \in E_1\}$
  $\alpha_{v_1} \leftarrow \sum_{x \in \mathrm{prev}} \phi_{(x,v_1)} \alpha_x$, $\alpha'_{v_1} \leftarrow \sum_{x \in \mathrm{prev}} \phi_{(x,v_1)} \left(h'_{(x,v_1)} \alpha_x + \alpha'_x\right)$
end for
$Z_1 \leftarrow \sum_{x \in \mathrm{snk}(G_1)} \alpha_x$
for $^\forall v_1 \in \mathrm{snk}(G_1)$, $\beta_{v_1} \leftarrow 1$, $\beta'_{v_1} \leftarrow 1$
for all $v_1 \in V_1$ in a reverse topological order **do**
  $\mathrm{next} \leftarrow \{x \in V_1 \mid (v_1, x) \in E_1\}$
  $\beta_{v_1} \leftarrow \sum_{x \in \mathrm{next}} \phi_{(v_1,x)} \beta_x$, $\beta'_{v_1} \leftarrow \sum_{x \in \mathrm{next}} \phi_{(v_1,x)} \left(h'_{(v_1,x)} \beta_x + \beta'_x\right)$
end for
for $^\forall k_1 \in \mathcal{K}_1$, $q_{k_1} \leftarrow 0$
for all $(u_1, v_1) \in E_1$ **do**
  $p \leftarrow \phi_{(u_1,v_1)} \left(\alpha_{u_1} \beta'_{v_1} + \alpha'_{u_1} \beta_{v_1}\right) / Z_1$
  for $^\forall k_1 \in \mathcal{K}_1$, $q_{k_1} \leftarrow q_{k_1} + p f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}$
end for

sentences and 47377 words. The number of the POS label types is equal to 45. The number of the label types used in base-phrase chunking is equal to 23.

We compare the proposed method to two existing sequence labeling methods as baselines. The POS labeler is the same in all the three methods used in this experiment. This labeler is a simple CRF and learned by ordinary optimization procedure. One baseline method is the 1-best pipeline method. A simple CRF model is learned for the chunking labeling, on the input sentences and the most likely POS label sequences predicted by the already learned POS labeler. We call this method "CRF + CRF." The other baseline method has a CRF model for the chunking labeling, which uses the marginalized features offered by the POS labeler. However, the parameters of the POS labeler are fixed in the training of the chunking model. This method corresponds to the method proposed in Bunescu (2008). We call this baseline "CRF + CRF-MF" ("MF" for "marginalized features"). The proposed method is the same as "CRF + CRF-MF", except that the both labelers are jointly trained by the procedure described in Section 3. We call this proposed method "CRF + CRF-BP" ("BP" for "back propagation").

	CRF + CRF	CRF + CRF-MF	CRF +CRF-BP
POS labeling	95.6	(95.6)	95.8
Base-phrase chunking	92.1	92.7	**93.1**

Table 2: Experimental result (F-measure)

In "CRF + CRF-BP," the objective function for joint learning (10) is not guaranteed to be convex, so optimization procedure is sensible to the initial configuration of the model parameters. In this experiment, we set the parameter values learned by "CRF + CRF-MF" as the initial values for the training of the "CRF + CRF-BP" method. Feature templates used in this experiment are listed in Table 1. Although we only described the formalization and optimization procedure of the models with arc features, We use node features in the experiment.

Table 2 shows the result of the methods we men-

=== Node feature templates ===
Node is source
Node is sink
Input word on the same time slice
Suffix of input word on the same time slice, n characters $(n \in [1, 2, 3])$
Initial word character is capitalized†
All word characters are capitalized†
Input word included in the vocabulary of POS $T^{\dagger}$ $(T \in \{(\text{All possible POS labels})\})$
Input word contains numbers†
POS label‡
=== Arc feature templates ===
Tail node is source
Head node is sink
Corresponding ordered pair of POS labels‡

Table 1: List of feature templates. All node features are combined with the corresponding node label (POS or chunking label) feature. All arc features are combined with the feature of the corresponding arc label pair. † features are instantiated on each time slice in five character window. ‡ features are not used in POS labeler, and marginalized as output features for "CRF + CRF-MF" and "CRF + CRF-BP."

tioned. In Table 2, bold numbers indicate significant improvement over the baseline models with $\alpha = 0.05$. From Table 2, the proposed method significantly outperforms two baseline methods on chunking performance. Although the improvement on POS labeling performance by the proposed method "CRF + CRF-BP" is not significant, it might show that optimization procedure provides some form of backward information propagation in comparison to "CRF + CRF-MF."

5 Conclusions

In this paper, we adopt the method to weight features on an upper sequence labeling stage by the marginalized probabilities estimated by the model on lower stages. We also point out that the model on an upper stage is considered to depend on the model on lower stages indirectly. In addition, we propose optimization procedure that enables the joint optimization of the multiple models on the different level of stages. We perform an experiment on a real-world task, and our method significantly outperforms existing methods.

We examined the effectiveness of the proposed method only on one task in comparison to just a few existing methods. In the future, we hope to compare our method to other competing methods like joint learning approaches in terms of both accuracy and computational efficiency, and perform extensive experiments on various tasks.

References

M. Berz. 1992. Automatic differentiation as nonarchimedean analysis. In *Computer Arithmetic and Enclosure*, pages 439–450.

R.C. Bunescu. 2008. Learning with probabilistic features for improved pipeline models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 670–679.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

G.F. Corliss, C. Faure, and A. Griewank. 2002. *Automatic differentiation of algorithms: from simulation to optimization*. Springer Verlag.

J.R. Finkel, C.D. Manning, and A.Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for seg-

menting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Z. Li and J. Eisner. 2009. First-and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 40–51.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):330.

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94. Cambridge MA, USA.

S. Sarawagi and W.W. Cohen. 2005. Semi-markov conditional random fields for information extraction. *Advances in Neural Information Processing Systems*, 17:1185–1192.

C. Sutton, A. McCallum, and K. Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*.

RE Wengert. 1964. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):464.

A Bayesian Mixture Model for Part-of-Speech Induction Using Multiple Features

Christos Christodoulopoulos
School of Informatics
University of Edinburgh
christos.c@ed.ac.uk

Sharon Goldwater
School of Informatics
University of Edinburgh
sgwater@inf.ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
steedman@inf.ed.ac.uk

Abstract

In this paper we present a fully unsupervised syntactic class induction system formulated as a Bayesian multinomial mixture model, where each word type is constrained to belong to a single class. By using a mixture model rather than a sequence model (e.g., HMM), we are able to easily add multiple kinds of features, including those at both the type level (morphology features) and token level (context and alignment features, the latter from parallel corpora). Using only context features, our system yields results comparable to state-of-the art, far better than a similar model without the one-class-per-type constraint. Using the additional features provides added benefit, and our final system outperforms the best published results on most of the 25 corpora tested.

1 Introduction

Research on unsupervised learning for NLP has become widespread recently, with part-of-speech induction, or syntactic class induction, being a particularly popular task.[1] However, despite a recent proliferation of syntactic class induction systems (Biemann, 2006; Goldwater and Griffiths, 2007; Johnson, 2007; Ravi and Knight, 2009; Berg-Kirkpatrick et al., 2010; Lee et al., 2010), careful comparison indicates that very few systems perform better than some much simpler and quicker methods dating back ten or even twenty years (Christodoulopoulos et al., 2010). This fact suggests that we should consider which features of the older systems led to their success, and attempt to combine these features with some of the machine learning methods introduced by the more recent systems. We pursue this strategy here, developing a system based on Bayesian methods where the probabilistic model incorporates several insights from previous work.

Perhaps the most important property of our model is that it is *type-based*, meaning that all tokens of a given word type are assigned to the same cluster. This property is not strictly true of linguistic data, but is a good approximation: as Lee et al. (2010) note, assigning each word type to its most frequent part of speech yields an upper bound accuracy of 93% or more for most languages. Since this is much better than the performance of current unsupervised syntactic class induction systems, constraining the model in this way seems likely to improve performance by reducing the number of parameters in the model and incorporating useful linguistic knowledge. Both of the older systems discussed by Christodoulopoulos et al. (2010), i.e., Clark (2003) and Brown et al. (1992), included this constraint and achieved very good performance relative to token-based systems. More recently, Lee et al. (2010) presented a new type-based model, and also reported very good results.

A second property of our model, which distinguishes it from the type-based Bayesian model of Lee et al. (2010), is that the underlying probabilistic model is a *clustering model*, (specifically, a multinomial mixture model) rather than a sequence model (HMM). In this sense, our model is more closely re-

[1] The task is more commonly referred to as part-of-speech induction, but we prefer the term syntactic class induction since the induced classes may not coincide with part-of-speech tags.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 638–647,
Edinburgh, Scotland, UK, July 27–31, 2011.

lated to several non-probabilistic systems that cluster context vectors or lower-dimensional representations of them (Redington et al., 1998; Schütze, 1995; Lamar et al., 2010). Sequence models are by far the most common method of supervised part-of-speech tagging, and have also been widely used for unsupervised part-of-speech tagging both with and without a dictionary (Smith and Eisner, 2005; Haghighi and Klein, 2006; Goldwater and Griffiths, 2007; Johnson, 2007; Ravi and Knight, 2009; Lee et al., 2010). However, systems based on context vectors have also performed well in these latter scenarios (Schütze, 1995; Lamar et al., 2010; Toutanova and Johnson, 2007) and present a viable alternative to sequence models.

One advantage of using a clustering model rather than a sequence model is that the features used for clustering need not be restricted to context words. Additional types of features can easily be incorporated into the model and inference procedure using the same general framework as in the basic model that uses only context word features. In particular, we present two extensions to the basic model. The first uses *morphological features*, which serve as cues to syntactic class and seemed to partly explain the success of two best-performing systems analysed by Christodoulopoulos et al. (2010). The second extension to our model uses *alignment features* gathered from parallel corpora. Previous work suggests that using parallel text can improve performance on various unsupervised NLP tasks (Naseem et al., 2009; Snyder and Barzilay, 2008).

We evaluate our model on 25 corpora in 20 languages that vary substantially in both syntax and morphology. As in previous work (Lee et al., 2010), we find that the one-class-per-type restriction boosts performance considerably over a comparable token-based model and yields results that are comparable to state-of-the-art even without the use of morphology or alignment features. Including morphology features yields the best published results on 14 or 15 of our 25 corpora (depending on the measure) and alignment features can improve results further.

2 Models

Our model is a multinomial mixture model with Bayesian priors over the mixing weights θ and multinomial class output parameters ϕ. The model is defined so that all observations associated with a single word type are generated from the same mixing component (syntactic class). In the basic model, these observations are token-level features; the morphology model adds type-level features as well. We begin by describing the simplest version of our model, where each word token is associated with a single feature, for example its left context word (the word that occurs to its left in the corpus). We then show how to generalise the model to multiple token-level features and to type-level features.

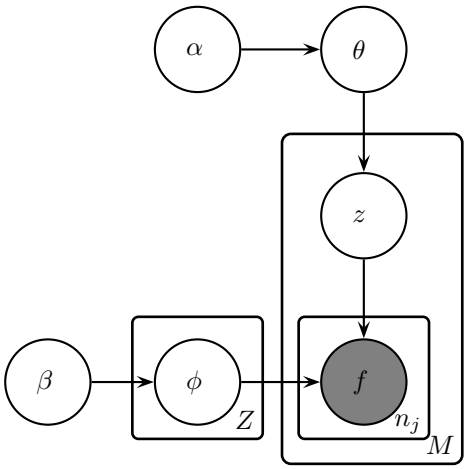


Figure 1: Plate diagram of the basic model with a single feature per token (the observed variable f). M, Z, and n_j are the number of word types, syntactic classes z, and features (= tokens) per word type, respectively.

2.1 Basic model

In the basic model, each word token is represented by a single feature such as its left context word. These features are the observed data; the model explains the data by assuming that it has been generated from some set of latent syntactic classes. The ith class is associated with a multinomial parameter vector ϕ_i that defines the distribution over features generated from that class, and with a mixing weight θ_i that defines the prior probability of that class. θ and ϕ_i are drawn from symmetric Dirichlet distributions with parameters α and β respectively.

The generative story goes as follows: First, generate the prior class probabilities θ. Next, for each

word type $j = 1 \ldots M$, choose a class assignment z_j from the distribution θ. For each class $i = 1 \ldots Z$, choose an output distribution over features ϕ_i. Finally, for each token $k = 1 \ldots n_j$ of word type j, generate a feature f_{jk} from ϕ_{z_j}, the distribution associated with the class that word type j is assigned to. The model is illustrated graphically in Figure 1 and is defined formally as follows:

$$\begin{aligned} \theta \,|\, \alpha &\sim \text{Dirichlet}(\alpha) \\ z_j \,|\, \theta &\sim \text{Multinomial}(\theta) \\ \phi_i \,|\, \beta &\sim \text{Dirichlet}(\beta) \\ f_{jk} \,|\, \phi_{z_j} &\sim \text{Multinomial}(\phi_{z_j}) \end{aligned}$$

In addition to the variables defined above, we will use F to refer to the number of different possible values a feature can take on (so that ϕ is a $Z \times F$ matrix). Thus, one way to think of the model is as a vector-based clustering system, where word type j is associated with a $1 \times F$ vector of feature counts representing the features of all n_j tokens of j, and these vectors are clustered into similar classes. The difference from other vector-based syntactic class induction systems is in the method of clustering. Here, we define a Gibbs sampler that samples from the posterior distribution of the clusters given the observed features; other systems have used various standard distance-based vector clustering methods. Some systems also include dimensionality reduction (Schütze, 1995; Lamar et al., 2010) to reduce the size of the context vectors; we simply use the F most common words as context features.

2.2 Inference

At inference time we want to sample a syntactic class assignment $\mathbf{z}$ from the posterior of the model. We use a collapsed Gibbs sampler, integrating out the parameters θ and ϕ and sampling from the following distribution:

$$P(\mathbf{z}|\mathbf{f}, \alpha, \beta) \propto P(\mathbf{z}|\alpha)P(\mathbf{f}|\mathbf{z}, \beta). \quad (1)$$

Rather than sampling the joint class assignment $P(\mathbf{z}|\mathbf{f}, \alpha, \beta)$ directly, the sampler iterates over each word type j, resampling its class assignment z_j given the current assignments $\mathbf{z}_{-j}$ of all other word types. The posterior over z_j can be computed as

$$\begin{aligned} &P(z_j \,|\, \mathbf{z}_{-j}, \mathbf{f}, \alpha, \beta) \\ &\quad \propto \; P(z_j \,|\, \mathbf{z}_{-j}, \alpha, \beta) P(\mathbf{f}_j \,|\, \mathbf{f}_{-j}, \mathbf{z}, \alpha, \beta) \quad (2) \end{aligned}$$

where $\mathbf{f}_j$ are the features associated with word type j (one feature for each token of j). The first (prior) factor is easy to compute due to the conjugacy between the Dirichlet and multinomial distributions, and is equal to

$$P(z_j = z \,|\, \mathbf{z}_{-j}, \alpha) = \frac{n_z + \alpha}{n_\cdot + Z\alpha} \quad (3)$$

where n_z is the number of types in class z and $n_\cdot$ is the total number of word types in all classes. All counts in this and the following equations are computed with respect to $\mathbf{z}_{-j}$ (e.g., $n_\cdot = M - 1$).

Computing the second (likelihood) factor is slightly more complex due to the dependencies between the different variables in $\mathbf{f}_j$ that are induced by integrating out the ϕ parameters. Consider first a simple case where word type j occurs exactly twice in the corpus, so $\mathbf{f}_j$ contains two features. The probability of the first feature f_{j1} is equal to

$$P(f_{j1} = f \,|\, z_j = z, \mathbf{z}_{-j}, \mathbf{f}_{-j}, \beta) = \frac{n_{f,z} + \beta}{n_{\cdot,z} + F\beta} \quad (4)$$

where $n_{f,z}$ is the number of times feature f has been seen in class z, $n_{\cdot,z}$ is the total number of feature tokens in the class, and F is the number of different possible features.

The probability of the second feature f_{j2} can be calculated similarly, except that it is conditioned on f_{j1} in addition to the other variables, so the counts for previously observed features must include the counts due to f_{j1} as well as those due to $\mathbf{f}_{-j}$. Thus, the probability is

$$\begin{aligned} &P(f_{j2} = f \,|\, f_{j1}, z_j = z, \mathbf{z}_{-j}, \mathbf{f}_{-j}, \beta) \\ &\qquad = \frac{n_{f,z} + \delta(f_{j1}, f_{j2}) + \beta}{n_{\cdot,z} + 1 + F\beta} \quad (5) \end{aligned}$$

where δ is the Kronecker delta function, equal to 1 if its arguments are equal and 0 otherwise.

Extending this example to the general case, the probability of a sequence of features $\mathbf{f}_j$ is computed using the chain rule, where the counts used in each factor are incremented as necessary for each additional conditioning feature, yielding the following expression:

$$\begin{aligned} &P(\mathbf{f}_j \,|\, \mathbf{f}_{-j}, z_j = z, \mathbf{z}_{-j}, \beta) \\ &\qquad = \frac{\prod_{k=1}^{F} \prod_{i=0}^{n_{jk}-1} (n_{jk,z} + i + \beta)}{\prod_{i=0}^{n_j-1} (n_{\cdot,z} + i + F\beta)} \quad (6) \end{aligned}$$

where n_{jk} is the number of instances of feature k in word type j.[2]

2.3 Extended models

We can extend the model above in two different ways: by adding more features at the word token level, or by adding features at the type level. To add more token-level features, we simply assume that each word token generates multiple features, one feature from each of several different kinds.[3] For example, the left context word might be one kind of feature and the right context word another. We assume conditional independence between the generated features given the syntactic class, so each kind of feature t has its own output parameters $\phi^{(t)}$. A plate diagram of the model with T kinds of features is shown in Figure 2 (a type-level feature is also included in this diagram, as described below).

Due to the independence assumption between the different kinds of features, the basic Gibbs sampler is easy to extend to this case by simpling multiplying in extra factors for the additional kinds of features, with the prior (Equation 3) unchanged. The likelihood becomes:

$$P(\mathbf{f}_j^{(1)}, \ldots, \mathbf{f}_j^{(T)} \mid \mathbf{f}_{-j}^{(1\ldots T)}, z_j = z, \mathbf{z}_{-j}, \beta) = \prod_{t=1}^{T} P(\mathbf{f}_j^{(t)} \mid \mathbf{f}_{-j}^{(t)}, z_j = z, \mathbf{z}_{-j}, \beta) \quad (7)$$

where each factor in the product is computed using Equation 6.

In addition to monolingual context features, we also explore the use of alignment features for those languages where we have parallel corpora. These features are extracted for language ℓ by word-aligning ℓ to another language ℓ' (details of the alignment procedure are described in Section 3.1). The features used for each token e in ℓ are the left and right context words of the word token that is aligned to e (if there is one). As with the monolingual context features, we use only the F most frequent words in ℓ' as possible features.

[2]One could approximate this likelihood term by assuming independence between all n_j feature tokens of word type j. This is the approach taken by Lee et al. (2010).

[3]We use the word *kind* here to avoid confusion with *type*, which we reserve for the type-token distinction, which can apply to features as well as words.

Note that this model with multiple context features is deficient: it can generate data that are inconsistent with any actual corpus, because there is no mechanism to constrain the left context word of token e_i to be the same as the right context word of token e_{i-1} (and similarly with alignment features). However, deficient models have proven useful in other unsupervised NLP tasks (Klein and Manning, 2002; Toutanova and Johnson, 2007). In particular, Toutanova and Johnson (2007) demonstrate good performance on unsupervised part-of-speech tagging (using a dictionary) with a Bayesian model similar to our own. If we remove the part of their model that relies on the dictionary (the morphological ambiguity classes), their model is equivalent to our own, without the restriction of one class per type. We use this token-based version of our model as a baseline in our experiments.

The final extension to our model introduces type-level features, specifically morphology features. The model is illustrated in Figure 2. We assume conditional independence between the morphology features and other features, so again we can simply multiply another factor into the likelihood during inference. There is only one morphological feature per type, so this factor has the form of Equation 4. Since frequent words will have many token-level features contributing to the likelihood and only one morphology feature, the morphology features will have a greater effect for infrequent words (as appropriate, since there is less evidence from context and alignments). As with the other kinds of features, we use only a limited number F_m of morphology features, as described below.

3 Experiments

3.1 Experimental setup

We evaluate our models using an increasing level of complexity, starting with a model that uses only monolingual context features. We use the $F = 100$ most frequent words as features, and consider two versions of this model: one with two kinds of features (one left and one right context word) and one with four (two context words on each side).

For the model with morphology features we ran the unsupervised morphological segmentation system Morfessor (Creutz and Lagus, 2005) to get a

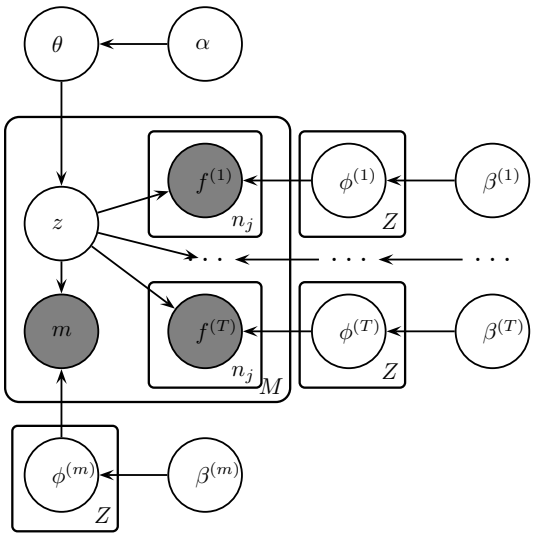


Figure 2: Plate diagram of the extended model with T kinds of token-level features ($f^{(t)}$ variables) and a single kind of type-level feature (morphology, m).

segmentation for each word type in the corpus. We then extracted the suffix of each word type[4] and used it as a feature type. This process yielded on average $F_m = 110$ morphological feature types[5]. Each word type generates at most one of these possible features. If there are overlapping possibilities (e.g. -ingly and -y) we take the longest possible match.

We also explore the idea of extending the morphology feature space beyond suffixes, by including features like capitalisation and punctuation. Specifically we use the features described in Haghighi and Klein (2006), namely *initial-capital*, *contains-hyphen*, *contains-digit* and we add an extra feature *contains-punctuation*.

For the model with alignment features, we follow (Naseem et al., 2009) in using only bidirectional alignments: using Giza++ (Och and Ney, 2003), we get the word alignments in both directions between all possible language pairs in our parallel corpora (i.e., alternating the source and target languages within each pair). We then use only those alignments that are found in both directions. As discussed above, we use two kinds of alignment features: the left and right context words of the aligned token in the other language. The feature space is set to the $F = 100$ most frequent words in that language.

Instead of fixing the hyperparameters α and β, we used the Metropolis-Hastings sampler presented by Goldwater and Griffiths (2007) to get updated values based on the likelihood of the data with respect to those hyperparameters[6]. In order to improve convergence of the sampler, we used simulated annealing with a sigmoid-shaped cooling schedule from an initial temperature of 2 down to 1. Preliminary experiments indicated that we could achieve better results by cooling even further (approximating the MAP solution rather than a sample from the posterior), so for all experiments reported here, we ran the sampler for a total of 2000 iterations, with the last 400 of these decreasing the temperature from 1 to 0.66.

Finally, we investigated two different initialisation techniques: First, we use random class assignments to word types (referred to as method 1) and second, we assign each of the Z most frequent word types to a separate class and then randomly distribute the rest of the word types to the classes (method 2).

3.2 Datasets

Although unsupervised systems should in principle be language- and corpus-independent, most part-of-speech induction systems (especially in the early literature) have been developed on English. Whether because English is simply an easier language, or because of bias introduced during development, these systems' performance is considerably worse in other languages (Christodoulopoulos et al., 2010)

Since we aim to use our system mostly on non-English corpora, and ones that are significantly smaller than the large English treebank corpora, we developed our models using one of the languages of the MULTEXT-East corpus (Erjavec, 2004), namely Bulgarian. The other languages in the corpus were used during development as a source of word alignments, but otherwise were only used for testing final versions of our models. Since none of the authors speak any of the languages in the MULTEXT col-

[4]Since Morfessor yields multiple affixes for each word we concatenated all the suffixes into a single suffix.

[5]There was large variance in the number of feature types for each language ranging from 11 in Chinese to more than 350 in German and Czech.

[6]For simplicity, we tied the β parameters for the two or four kinds of context features to the same value, and similarly the β parameters for the two kinds of alignment features.

lection, we also used the Penn Treebank WSJ corpus (Marcus et al., 1993) for development. Following Christodoulopoulos et al. (2010) we created a smaller version of the WSJ corpus (referred to as wsj-s) to approximate the size of the corpora in MULTEXT-East. For comparison to other systems, we also used the full WSJ at test time.

For further testing, we used the remaining MULTEXT languages, as well as the languages of the CONNL-X (Buchholz and Marsi, 2006) shared task. This dataset contains 13 languages, 4 of which are freely available (Danish, Dutch, Portuguese and Swedish) and 9 that are used with permission from the creators of the corpora (Arabic[7], Bulgarian[8], Czech[9], German[10], Chinese[11], Japanese[12], Slovene[13], Spanish[14], Turkish[15]). Following Lee et al. (2010) we used only the training sections for each language.

Finally, to widen the scope of our system, we generated two more corpora in French[16] and Ancient Greek[17], extracting the gold standard parts of speech from the respective dependency treebanks.

3.3 Baselines

We chose three baselines for comparison. The first is the basic k-means clustering algorithm, which we applied to the same feature vectors we extracted for our system (context + extended morphology), using a Euclidean distance metric. This provides a very simple vector-based clustering baseline. The second baseline is a more recent vector-based syntactic class induction method, the SVD approach of (Lamar et al., 2010), which extends Schütze (1995)'s original method and, like ours, enforces a one-class-per-tag restriction. As a third baseline we use the system of Clark (2003) since it is a type-level system that models morphology and has produced very good results on multilingual corpora.

[7]Part of the Prague Arabic Treebank (Hajič et al., 2003; Smrž and Pajas, 2004)

[8]Part of the BulTreeBank (Simov et al., 2004).

[9]Part of the Prague Dep. Treebank (Böhmová et al., 2001)

[10]Part of the TIGER Treebank (Brants et al., 2002)

[11]Part of the Sinica Treebank (Keh-Jiann et al., 2003)

[12]Part of the Tübingen Treebank of Spoken Japanese (formerly VERMOBIL Treebank - Kawata and Bartels (2000)).

[13]Part of the Slovene Dep. Treebank (Džeroski et al., 2006)

[14]Part of the Cast3LB Treebank (Civit et al., 2006)

[15]Part of the METU-Sabanci Treebank (Oflazer et al., 2003).

[16]French Treebank (Abeillé et al., 2000)

[17]Greek Dependency Treebank (Bamman et al., 2009)

4 Results and Analysis

4.1 Development results

Tables 1 and 2 present the results from development runs, which were used to decide which features to incorporate in the final system. We used V-Measure (Rosenberg and Hirschberg, 2007) as our primary evaluation score, but also present many-to-one matching accuracy (M-1) scores for better comparison with previously published results. We chose V-Measure (VM) as our evaluation score because it is less sensitive to the number of classes induced by the model (Christodoulopoulos et al., 2010), allowing us to develop our models without using the number of classes as a parameter. We fixed the number of classes in all systems to 45 during development; note however that the gold standard tag set for Bulgarian contains only 12 tags, so the results in Table 1 (especially the M-1 scores) are not comparable to previous results. For results using the number of gold-standard tags refer to Table 4.

The first conclusion that can be drawn from these results is the large difference between the token- and type-based versions of our system, which confirms that the one-class-per-type restriction is helpful for unsupervised syntactic class induction. We also see that for both languages, the performance of the model using 4 context words (± 2 on each side) is worse than the 2 context words model. We therefore used only two context words for all of our additional test languages (below).

We can clearly see that morphological features are helpful in both languages; however the extended features of Haghighi and Klein (2006) seem to help only on the English data. This could be due to the fact that Bulgarian has a much richer morphology and thus the extra features contribute little to the overall performance of the model.

The contribution of the alignment features on the Bulgarian corpus (aligned with English) is less significant than that of morphology but when combined, the two sets of features yield the best performance. This provides evidence in favor of using multiple features.

Finally, initialisation method 2 does not yield

system	±1 words VM/M-1	±2 words VM/M-1
base	58.1 / 70.8	55.4 / 67.6
base(tokens)	48.3 / 62.5	37.0 / 54.4
base(init)	57.6 / 70.1	56.1 / 68.6
+morph	58.3 / 74.9	57.4 / 71.9
+morph(ext)	57.8 / 73.7	57.8 / 70.1
(init)+morph	57.8 / 74.3	57.3 / 69.5
(init)+morph(ext)	58.1 / 74.3	57.2 / 71.3
+aligns(EN)	58.1 / 72.6	56.7 / 71.1
+aligns(EN)+morph	**59.0 / 75.4**	57.5 / 69.7

Table 1: V-measure (VM) and many-to-one (M-1) results on the MULTEXT-Bulgarian corpus for various models using either ±1 or ±2 context words as features. base: context features only; (tokens): token-based model; (init): Initialisation method 2—other results use method 1; (ext): Extended morphological features.

system	±1 words VM/M-1	±2 words VM/M-1
base	63.3 / 64.3	62.4 / 63.3
base(tokens)	48.6 / 57.8	49.3 / 38.3
base(init)	62.7 / 62.9	62.2 / 62.4
+morph	66.4 / 66.7	65.1 / 67.2
+morph(ext)	**67.7 / 72.0**	65.6 / 67.0
(init)+morph	64.8 / 66.9	64.2 / 66.0
(init)+morph(ext)	67.4 / 71.3	65.7 / 67.1

Table 2: V-measure and many-to-one results on the wsj-s corpus for various models, as described in Table 1.

.

consistent improvements over the standard random initialisation—if anything, it seems to perform worse. We therefore use only method 1 in the remaining experiments.

4.2 Overall results

Table 3 presents the results on our parallel corpora. We tested all possible combinations of two languages to align, and present both the average score over all alignments, and the score under the best choice of aligned language.[18] Also shown are the results of adding morphology features to the basic model (context features only) and to the best alignment model for each language. In accord with our development results, adding morphology to the basic model is generally useful. The alignment results are mixed: on the one hand, choosing the best possible language to align yields improvements, which can be improved further by adding morphological features, resulting in the best scores of all models for most languages. On the other hand, without knowing which language to choose, alignment features do not help on average. We note, however, that three out of the seven languages have English as their best-aligned pair (perhaps due to its better overall scores), which suggests that in the absence of other knowledge, aligning with English may be a good choice.

The low average performance of the alignment features is disappointing, but there are many possible variations on our method for extracting these features that we have not yet tested. For example, we used only bidirectional alignments in an effort to improve alignment precision, but these alignments typically cover less than 40% of tokens. It is possible that a higher-recall set of alignments could be more useful.

We turn now to our results on all 25 corpora, shown in Table 4 along with corpus statistics, baseline results, and the best published results for each language (when available). Our system, including morphology features in all cases, is listed as BMMM (Bayesian Multinomial Mixture Model). We do not include alignment features for the MULTEXT languages since these features only yielded improvements for the oracle case where we know which aligned language to choose. Nevertheless, our MULTEXT scores mostly outperform all other systems. Overall, we acheive the highest published results on 14 (VM) or 15 (M-1) of the 25 corpora.

One surprising discovery is the high performance of the k-means clustering system. Despite its simplicity, it is competitive with the other systems and in a few cases even achieves the best published results.

5 Conclusion

We have presented a Bayesian model for syntactic class induction that has two important properties. First, it is type-based, assigning the same class to every token of a word type. We have shown by

[18] The choice of language was based on the same test data, so the 'best-language' results should be viewed as oracle scores.

Lang.	BASE		ALIGNMENTS		
	base	**+morph**	**Avg.**	**Best**	**+morph**
	VM/M-1	VM/M-1	VM/M-1	VM/M1	VM/M1
Bulgarian	54.4 / 61.5	54.5 / 64.3	53.1 / 60.5	55.2 / 64.5(EN)	**55.7 / 66.0**
Czech	54.2 / 58.9	53.9 / 64.2	52.6 / 58.4	53.8 / 59.7(EN)	**55.4 / 66.4**
English	62.9 / 72.4	63.3 / 73.3	62.5 / 72.0	63.2 / 71.9(HU)	**63.5 / 73.7**
Estonian	52.8 / 63.5	53.3 / **67.4**	52.8 / 63.9	53.5 / 65.0(EN)	**54.3** / 66.9
Hungarian	53.3 / 60.4	54.8 / **68.2**	53.3 / 60.8	53.9 / 61.1(RO)	**55.9** / 67.1
Romanian	53.9 / 62.4	52.3 / 61.1	56.2 / 63.7	**57.5 / 64.6**(ES)	54.5 / 63.4
Slovene	**57.2** / 65.9	56.7 / 67.9	54.7 / 64.1	55.9 / 64.4(HU)	56.7 / **67.9**
Serbian	**49.1** / 56.6	49.0 / **62.0**	47.3 / 55.6	48.9 / 59.4(CZ)	48.3 / 60.8

Table 3: V-measure (VM) and many-to-one (M-1) results on the languages in the MULTEXT-East corpus using the gold standard number of classes shown in Table 4. BASE results use $\pm$1-word context features alone or with morphology. ALIGNMENTS adds alignment features, reporting the average score across all possible choices of paired language and the scores under the best performing paired language (in parens), alone or with morphology features.

	Language	Types	Tags	k-means	SVD2	clark	Best Pub.	BMMM
WSJ	wsj	49,190	45	59.5 / 61.6	58.2 / 64.0	65.6 / 71.2	68.8 / 76.1*	66.1 / 72.8
	wsj-s	16,850	45	56.7 / 60.1	54.3 / 60.7	63.8 / 68.8	62.3 / 70.7*	**67.7 / 72.0**
MULTEXT-East	Bulgarian	16,352	12	50.3 / 59.3	41.7 / 51.0	**55.6 / 66.5**	-	54.5 / 64.4
	Czech	19,115	12	48.6 / 56.7	35.5 / 50.9	52.6 / 64.1	-	**53.9 / 64.2**
	English	9,773	12	56.5 / 65.4	52.3 / 65.5	60.5 / 70.6	-	**63.3 / 73.3**
	Estonian	17,845	11	45.3 / 55.6	38.7 / 55.3	44.4 / 58.4	-	**53.3 / 64.4**
	Hungarian	20,321	12	46.7 / 53.9	39.8 / 49.5	48.9 / 61.4	-	**54.8 / 68.2**
	Romanian	15,189	14	45.2 / 55.1	42.1 / 52.6	40.9 / 49.9	-	**52.3 / 61.1**
	Slovene	17,871	12	46.9 / 56.2	39.5 / 54.2	54.9 / 69.4	-	**56.7 / 67.9**
	Serbian	18,095	12	41.4 / 47.0	39.1 / 54.6	**51.0 / 64.1**	-	49.0 / 62.0
CoNLL06 Shared Task	Arabic	12,915	20	**43.3** / 60.7	27.6 / 49.0	40.6 / 59.8	-	42.4 / **61.5**
	Bulgarian	32,439	54	53.6 / 65.6	49.0 / 65.3	**59.6 / 70.4**	-	58.8 / 68.9
	Chinese	40,562	15	32.6 / 61.1	24.5 / 54.6	31.8 / 56.7	-	**42.6 / 69.4**
	Czech	130,208	12	-	-	47.1 / 65.5	-	**48.4 / 65.7**
	Danish	18,356	25	51.7 / 61.6	40.8 / 57.6	52.7 / 65.3	- / 66.7†	**59.0 / 71.1**
	Dutch	28,393	13	45.3 / 60.5	36.7 / 52.4	52.2 / 67.9	- / 67.3‡	**54.7 / 71.1**
	German	72,326	54	58.7 / 67.5	54.1 / 64.2	**63.0** / 73.9	- / 68.4‡	61.9 / **74.4**
	Japanese	3,231	80	76.1 / 76.2	74.4 / 75.5	**78.6** / 77.4	-	77.4 / **78.5**
	Portuguese	28,931	22	51.6 / 64.4	45.9 / 63.1	57.4 / 69.2	- / 75.3†	**63.9 / 76.8**
	Slovene	7,128	29	52.6 / **64.2**	44.0 / 60.3	**53.9** / 63.5	-	49.4 / 56.2
	Spanish	16,458	47	59.5 / 69.2	54.8 / 68.2	61.6 / 71.9	- / **73.2**†	**63.2** / 71.7
	Swedish	20,057	41	53.2 / 62.2	47.4 / 59.1	**58.9 / 68.7**	- / 60.6‡	58.0 / 68.2
	Turkish	17,563	30	**40.8 / 62.8**	27.4 / 52.4	36.8 / 58.1	-	40.2 / 58.7
	French	49,964	23	48.2 / 68.6	46.3 / 68.5	**57.3 / 77.8**	-	55.0 / 76.6
	A.Greek	15,194	15	38.6 / 44.8	24.2 / 38.5	33.3 / **45.4**	-	**40.5** / 45.1

Table 4: Final results on 25 corpora in 20 languages, with the number of induced classes equal to the number of gold standard tags in all cases. **k-means** and **SVD2** models could not produce a clustering in the Czech CoNLL corpus due its size. Best published results are from *Christodoulopoulos et al. (2010), †Berg-Kirkpatrick et al. (2010) and ‡Lee et al. (2010). The latter two papers do not report VM scores. No best published results are shown for the MULTEXT languages; Christodoulopoulos et al. (2010) report results based on 45 tags suggesting that **clark** performs best on these corpora.

comparison with a token-based version of the model that this restriction is very helpful. Second, it is a clustering model rather than a sequence model. This property makes it easy to incorporate multiple kinds of features into the model at either the token or the type level. Here, we experimented with token-level context features and alignment features and type-level morphology features, showing that morphology features are helpful in nearly all cases, and alignment features can be helpful if the aligned language is properly chosen. Our results even without these extra features are competitive with state-of-the-art; with the additional features we achieve the best published results in the majority of the 25 corpora tested.

Since it is so easy to add extra features to our model, one direction for future work is to explore other possible features. For example, it could be useful to add dependency features from an unsupervised dependency parser. We are also interested in improving our morphology features, either by considering other ways to extract features during preprocessing (for example, including prefixes or not concatenating together all suffixes), or by developing a joint model for inducing both morphology and syntactic classes simultaneously. Finally, our model could be extended by replacing the standard mixture model with an infinite mixture model (Rasmussen, 2000) in order to induce the number of syntactic classes automatically.

Acknowledgments

The authors would like to thank Emily Thomforde, Ioannis Konstas, Tom Kwiatkowski and the anonymous reviewers for their comments and suggestions. We would also like to thank Kiril Simov, Toni Marti, Tomaz Erjavec, Jess Lin and Kathrin Beck for providing us with CoNLL data. This work was supported by an EPSRC graduate Fellowship, and by ERC Advanced Fellowship 249520 GRAMPLUS.

References

Anne Abeillé, Lionel Clément, and Alexandra Kinyon. 2000. Building a treebank for French. In *In Proceedings of the LREC 2000*.

David Bamman, Francesco Mambrini, and Gregory Crane. 2009. An ownership model of annotation: The Ancient Greek dependency treebank. In *TLT 2009-Eighth International Workshop on Treebanks and Linguistic Theories*.

Taylor Berg-Kirkpatrick, Alexandre B. Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL 2010*, pages 582–590, Los Angeles, California, June.

Chris Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of COLING ACL 2006*, pages 7–12, Morristown, NJ, USA.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2001. The Prague dependency treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*, pages 103 – 126. Kluwer Academic Publishers.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

Peter F. Brown, Vincent J. Della Pietra, Peter V. Desouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA, October. Association for Computational Linguistics.

Montserrat Civit, Ma. Martí, and Núria Bufí. 2006. Cat3lb and cast3lb: From constituents to dependencies. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *Advances in Natural Language Processing*, volume 4139 of *Lecture Notes in Computer Science*, pages 141–152. Springer Berlin / Heidelberg.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of EACL 2003*, pages 59–66, Morristown, NJ, USA.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *In Proceedings of the International and Interdisciplinary Conference on*

Adaptive Knowledge Representation and Reasoning (AKRR'05), volume 5, pages 106–113.

Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdenek Žabokrtsky, and Andreja Žele. 2006. Towards a Slovene dependency treebank. In *Proceedings Int. Conf. on Language Resources and Evaluation (LREC)*.

Tomaž Erjavec. 2004. MULTEXT-East version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *Fourth International Conference on Language Resources and Evaluation, (LREC'04)*, pages 1535 – 1538, Paris. ELRA.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of ACL 2007*, pages 744–751, Prague, Czech Republic, June.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of NAACL 2006*, pages 320–327, Morristown, NJ, USA.

Jan Hajič, Jarmila Panevová, Zdeňka Urešová, Alevtina Bémová, and Petr Pajas. 2003. PDTVALLEX: creating a large-coverage valency lexicon for treebank annotation. In *Proceedings of The Second Workshop on Treebanks and Linguistic Theories*, pages 57–68. Vaxjo University Press.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of EMNLP-CoNLL 2007*, pages 296–305, Prague, Czech Republic, June.

Yasushira Kawata and Julia Bartels. 2000. Stylebook for the Japanese treebank in VERMOBIL. Technical report, Universität Tüubingen.

Chen Keh-Jiann, Chu-Ren Huang, Feng-Yi Chen, Chi-Ching Luo, Ming-Chung Chang, Chao-Jan Chen, and Zhao-Ming Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*, pages 231–248. Kluwer Academic Publishers.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL 40*, pages 128–135.

Michael Lamar, Yariv Maron, Mark Johnson, and Elie Bienenstock. 2010. SVD and clustering for unsupervised POS tagging. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 215–219, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2010. Simple type-level unsupervised POS tagging. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 853–861, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):331–330.

Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. 2009. Multilingual Part-of-Speech tagging: Two unsupervised approaches. *Journal of Artificial Intelligence Research*, 36:341–385.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51.

Kemal Oflazer, Bilge Say, Dilek Z. Hakkani-Tür, and Gökhan Tür, 2003. *Building A Turkish Treebank*, chapter 1, pages 1–17. Kluwer Academic Publishers.

Carl Rasmussen. 2000. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems 12*.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised Part-of-Speech tagging. In *Proceedings of ACL-IJCNLP 2009*, pages 504–512, Suntec, Singapore, August.

Martin Redington, Nick Chater, and Steven Finch. 1998. Distributional information: a powerful cue for acquiring syntactic categories. *Cognitive Science*, 22:425 – 469.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of EMNLP-CoNLL 2007*, pages 410–420.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of EACL 7*, pages 141–148, San Francisco, CA, USA.

Kiril Simov, Petya Osenova, Alexander Simov, and Milen Kouylekov. 2004. Design and implementation of the Bulgarian HPSG-based treebank. *Research on Language & Computation*, 2(4):495–522.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *Proceedings of ACL 2005*, pages 354–362, Morristown, NJ, USA.

Otakar Smrž and Petr Pajas. 2004. Morphotrees of Arabic and their annotation in the TrEd environment. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*, pages 38–41.

Ben Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL*.

Kristina Toutanova and Mark Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of NIPS 2007*.

Large-Scale Noun Compound Interpretation Using Bootstrapping and the Web as a Corpus

Su Nam Kim
Computer Science & Software Engineering
University of Melbourne
Melbourne, VIC 3010
Australia
snkim@csse.unimelb.edu.au

Preslav Nakov
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
nakov@comp.nus.edu.sg

Abstract

Responding to the need for semantic lexical resources in natural language processing applications, we examine methods to acquire noun compounds (NCs), e.g., *orange juice*, together with suitable fine-grained semantic interpretations, e.g., *squeezed from*, which are directly usable as paraphrases. We employ bootstrapping and web statistics, and utilize the relationship between NCs and paraphrasing patterns to jointly extract NCs and such patterns in multiple alternating iterations. In evaluation, we found that having one compound noun fixed yields both a higher number of semantically interpreted NCs and improved accuracy due to stronger semantic restrictions.

1 Introduction

Noun compounds (NCs) such as *malaria mosquito* and *colon cancer tumor suppressor protein* are challenging for text processing since the relationship between the nouns they are composed of is implicit. NCs are abundant in English and understanding their semantics is important in many natural language processing (NLP) applications. For example, a question answering system might need to know whether *protein acting as a tumor suppressor* is a good paraphrase for *tumor suppressor protein*. Similarly, a machine translation system facing the unknown noun compound *Geneva headquarters* might translate it better if it could first paraphrase it as *Geneva headquarters of the WTO*. Given a query for *"migraine treatment"*, an information retrieval system could use paraphrasing verbs like *relieve* and *prevent* for query expansion and result ranking.

Most work on noun compound interpretation has focused on two-word NCs. There have been two general lines of research: the first one derives the NC semantics from the semantics of the nouns it is made of (Rosario and Hearst, 2002; Moldovan et al., 2004; Kim and Baldwin, 2005; Girju, 2007; Séaghdha, 2009; Tratz and Hovy, 2010), while the second one models the relationship between the nouns directly (Vanderwende, 1994; Lapata, 2002; Kim and Baldwin, 2006; Nakov and Hearst, 2006; Nakov and Hearst, 2008; Butnariu and Veale, 2008).

In either case, the semantics of an NC is typically expressed by an abstract relation like CAUSE (e.g., *malaria mosquito*), SOURCE (e.g., *olive oil*), or PURPOSE (e.g., *migraine drug*), coming from a small fixed inventory. Some researchers however, have argued for a more fine-grained, even infinite, inventory (Finin, 1980). Verbs are particularly useful in this respect and can capture elements of the semantics that the abstract relations cannot. For example, while most NCs expressing MAKE, can be paraphrased by common patterns like *be made of* and *be composed of*, some NCs allow more specific patterns, e.g., *be squeezed from* for *orange juice*, and *be topped with* for *bacon pizza*.

Recently, the idea of using fine-grained paraphrasing verbs for NC semantics has been gaining popularity (Butnariu and Veale, 2008; Nakov, 2008b); there has also been a related shared task at SemEval-2010 (Butnariu et al., 2010). This interest is partly driven by practicality: verbs are directly usable as paraphrases. Still, abstract relations remain dominant since they offer a more natural generalization, which is useful for many NLP applications.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 648–658,
Edinburgh, Scotland, UK, July 27–31, 2011.

One good contribution to this debate would be a direct study of the relationship between fine-grained and coarse-grained relations for NC interpretation. Unfortunately, the existing datasets do not allow this since they are tied to one particular granularity; moreover, they only contain a few hundred NCs. Thus, our objective is to build a large-scale dataset of hundreds of thousands of NCs, each interpreted (1) by an abstract semantic relation and (2) by a set of paraphrasing verbs. Having such a large dataset would also help the overall advancement of the field.

Since there is no universally accepted abstract relation inventory in NLP, and since we are interested in NC semantics from both a theoretical and a practical viewpoint, we chose the set of abstract relations proposed in the theory of Levi (1978), which is dominant in theoretical linguistics and has been also used in NLP (Nakov and Hearst, 2008).

We use a two-step algorithm to jointly harvest NCs and patterns (verbs and prepositions) that interpret them for a given abstract relation. First, we extract NCs using a small number of seed patterns from a given abstract relation. Then, using the extracted NCs, we harvest more patterns. This is repeated until no new NCs and patterns can be extracted or for a pre-specified number of iterations. Our approach combines pattern-based extraction and bootstrapping, which is novel for NC interpretation; however, such combinations have been used in other areas, e.g., named entity recognition (Riloff and Jones, 1999; Thelen and Riloff, 2002; Curran et al., 2007; McIntosh and Curran, 2009).

The remainder of the paper is organized as follows: Section 2 gives an overview of related work, Section 3 motivates our semantic representation, Sections 4, 5, and 6 explain our method, dataset and experiments, respectively, Section 7 discusses the results, Section 8 provides error analysis, and Section 9 concludes with suggestions for future work.

2 Related Work

As we mentioned above, the implicit relation between the two nouns forming a noun compound can often be expressed overtly using verbal and prepositional paraphrases. For example, *student loan* is "loan given to a student", while *morning tea* can be paraphrased as "tea in the morning".

Thus, many NLP approaches to NC semantics have used verbs and prepositions as a fine-grained semantic representation or as features when predicting coarse-grained abstract relations. For example, Vanderwende (1994) associated verbs extracted from definitions in an online dictionary with abstract relations. Lauer (1995) expressed NC semantics using eight prepositions. Kim and Baldwin (2006) predicted abstract relations using verbs as features. Nakov and Hearst (2008) proposed a fine-grained NC interpretation using a distribution over Web-derived verbs, prepositions and coordinating conjunctions; they also used this distribution to predict coarse-grained abstract relations. Butnariu and Veale (2008) adopted a similar fine-grained verb-centered approach to NC semantics. Using a distribution over verbs as a semantic interpretation was also carried out in a recent challenge: SemEval-2010 Task 9 (Butnariu et al., 2009; Butnariu et al., 2010).

In noun compound interpretation, verbs and prepositions can be seen as patterns connecting the two nouns in a paraphrase. Similar pattern-based approaches have been popular in information extraction and ontology learning. For example, Hearst (1992) extracted hyponyms using patterns such as *X, Y, and/or other Zs*, where *Z* is a hypernym of *X* and *Y*. Berland and Charniak (1999) used similar patterns to extract meronymy (part-whole) relations, e.g., *parts/NNS of/IN wholes/NNS* matches *basements of buildings*. Unfortunately, matches are rare, which makes it difficult to build large semantic inventories. In order to overcome data sparseness, pattern-based approaches are often combined with bootstrapping. For example, Riloff and Jones (1999) used a multi-level bootstrapping algorithm to learn both a semantic lexicon and extraction patterns, e.g., *owned by X* extracts COMPANY and *facilities in X* extracts LOCATION. That is, they learned semantic lexicons using extraction patterns, and then, alternatively, they extracted new patterns using these lexicons. They also introduced a second level of bootstrapping to retain the most reliable examples only. While the method enables the extraction of large lexicons, its quality degrades rapidly, which makes it impossible to run for too many iterations. Recently, Curran et al. (2007) and McIntosh and Curran (2009) proposed ways to control degradation using simultaneous learning and weighting.

Bootstrapping has been applied to noun compound extraction as well. For example, Kim and Baldwin (2007) used it to produce a large number of semantically interpreted noun compounds from a small number of seeds. In each iteration, the method replaced one component of an NC with its synonyms, hypernyms and hyponyms to generate a new NC. These new NCs were further filtered based on their semantic similarity with the original NC. While the method acquired a large number of noun compounds without significant semantic drifting, its accuracy degraded rapidly after each iteration. More importantly, the variation of the sense pairs was limited since new NCs had to be semantically similar to the original NCs.

Recently, Kozareva and Hovy (2010) combined patterns and bootstrapping to learn the selectional restrictions for various semantic relations. They used patterns involving the coordinating conjunction *and*, e.g., "* *and John fly to* *", and learned arguments such as *Mary/Tom* and *France/New York*. Unlike in NC interpretation, it is not necessary for their arguments to form an NC, e.g., *Mary France* and *France Mary* are not NCs. Rather, they were interested in building a semantic ontology with a predefined set of semantic relations, similar to YAGO (Suchanek et al., 2007), where the pattern *work for* would have arguments like *a company/UNICEF*.

3 Semantic Representation

Inspired by (Finin, 1980), Nakov and Hearst (2006) and (Nakov, 2008b) proposed that NC semantics is best expressible using paraphrases involving verbs and/or prepositions. For example, *bronze statue* is a statue that *is made of, is composed of, consists of, contains, is of, is, is handcrafted from, is dipped in, looks like* bronze. They further proposed that selecting one such paraphrase is not enough and that multiple paraphrases are needed for a fine-grained representation. Finally, they observed that not all paraphrases are equally good (e.g., *is made of* is arguably better than *looks like* or *is dipped in* for MAKE), and thus proposed that the semantics of a noun compound should be expressed as a *distribution* over multiple possible paraphrases. This line of research was later adopted by SemEval-2010 Task 9 (Butnariu et al., 2010).

It easily follows that the semantics of abstract relations such as MAKE that can hold between the nouns in an NC can be represented in the same way: as a distribution over paraphrasing verbs and prepositions. Note, however, that some NCs are paraphrasable by more specific verbs that do not necessarily support the target abstract relation. For example, *malaria mosquito*, which expresses CAUSE, can be paraphrased using verbs like *carry*, which do not imply direct causation. Thus, while we will be focusing on extracting NCs for a particular abstract relation, we are interested in building semantic representations that are specific for these NCs and do not necessarily apply to *all* instances of that relation.

Traditionally, the semantics of a noun compound have been represented as an abstract relation drawn from a small closed set. Unfortunately, no such set is universally accepted, and mapping between sets has proven challenging (Girju et al., 2005). Moreover, being both abstract and limited, such sets capture only part of the semantics; often multiple meanings are possible, and sometimes none of the pre-defined ones suits a given example. Finally, it is unclear how useful these sets are since researchers have often fallen short of demonstrating practical uses.

Arguably, verbs have more expressive power and are more suitable for semantic representation: there is an infinite number of them (Downing, 1977), and they can capture fine-grained aspects of the meaning. For example, while both *wrinkle treatment* and *migraine treatment* express the same abstract relation TREATMENT-FOR-DISEASE, fine-grained differences can be revealed using verbs, e.g., *smooth* can paraphrase the former, but not the latter.

In many theories, verbs play an important role in NC derivation (Levi, 1978). Moreover, speakers often use verbs to make the hidden relation between the noun in a noun compound overt. This allows for simple extraction and for straightforward use in NLP tasks like textual entailment (Tatu and Moldovan, 2005) and machine translation (Nakov, 2008a).

Finally, a single verb is often not enough, and the meaning is better approximated by a collection of verbs. For example, while *malaria mosquito* expresses CAUSE (and is paraphrasable using *cause*), further aspects of the meaning can be captured with more verbs, e.g., *carry*, *spread*, *be responsible for*, *be infected with*, *transmit*, *pass on*, etc.

4 Method

We harvest noun compounds expressing some target abstract semantic relation (in the experiments below, this is Levi's MAKE$_2$), starting from a small number of initial seed patterns: paraphrasing verbs and/or prepositions. Optionally, we might also be given a small number of noun compounds that instantiate the target abstract relation. We then learn more noun compounds and patterns for the relation by alternating between the following two bootstrapping steps, using the Web as a corpus. First, we extract more noun compounds that are paraphrasable with the available patterns (see Section 4.1). We then look for new patterns that can paraphrase the newly-extracted noun compounds (see Section 4.2). These two steps are repeated until no new noun compounds can be extracted or until a pre-determined number of iterations has been reached. A schematic description of the algorithm is shown in Figure 1.

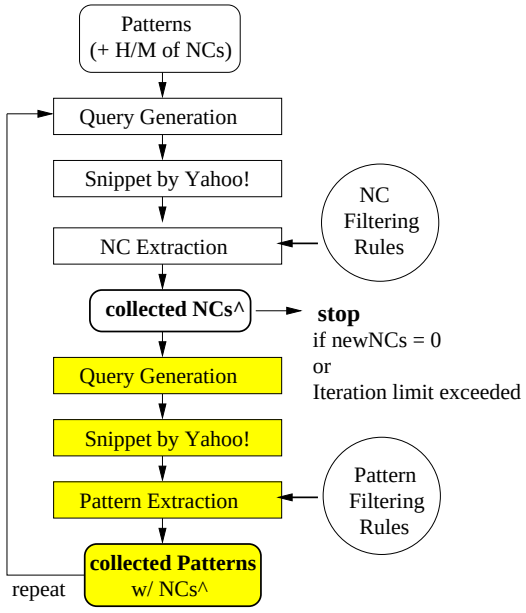


Figure 1: Our bootstrapping algorithm.

4.1 Bootstrapping Step 1: Noun Compound Extraction

Given a list of patterns (verbs and/or prepositions), we mine the Web to extract noun compounds that match these patterns. We experiment with the following three bootstrapping strategies for this step:

- **Loose bootstrapping** uses the available patterns and imposes no further restrictions.
- **Strict bootstrapping** requires that, in addition to the patterns themselves, some noun compounds matching each pattern be made available as well. A pattern is only instantiated in the context of either the head or the modifier of a noun compound that is known to match it.
- **NC-only strict bootstrapping** is a stricter version of *strict bootstrapping*, where the list of patterns is limited to the initial seeds.

Below we describe each of the sub-steps of the NC extraction process: query generation, snippet harvesting, and noun compound acquisition & filtering.

4.1.1 Query Generation

We generate generalized exact-phrase queries to be used in a Web search engine (we use *Yahoo!*):

`"* that PATTERN *"`	(*loose*)
`"HEAD that PATTERN *"`	(*strict*)
`"* that PATTERN MOD"`	(*strict*)

where `PATTERN` is an inflected form of a verb, `MOD` and `HEAD` are inflected forms the modifier and the head of a noun compound that is paraphrasable by the pattern, `that` is the word *that*, and `*` is the search engine's star operator.

We use the first pattern for *loose bootstrapping* and the other two for both *strict bootstrapping* and *NC-only strict bootstrapping*.

Note that the above queries are generalizations of the actual queries we use against the search engine. In order to instantiate these generalizations, we further generate the possible inflections for the verbs and the nouns involved. For nouns, we produce singular and plural forms, while for verbs, we vary not only the number (singular and plural), but also the tense (we allow present, past, and present perfect). When inflecting verbs, we distinguish between active verb forms like *consist of* and passive ones like *be made from* and we treat them accordingly. Overall, in the case of *loose bootstrapping*, we generate about 14 and 20 queries per pattern for active and passive patterns, respectively, while for *strict bootstrapping* and *NC-only strict bootstrapping*, the instantiations yield about 28 and 40 queries for active and passive patterns, respectively.

For example, given the seed *be made of*, we could generate "⋆ that were made of ⋆". If we are further given the NC *orange juice*, we could also produce "juice that was made of ⋆" and "⋆ that is made of oranges".

4.1.2 Snippet Extraction

We execute the above-described instantiations of the generalized queries against a search engine as exact phrase queries, and, for each one, we collect the snippets for the top 1,000 returned results.

4.1.3 NC Extraction and Filtering

Next, we process the snippets returned by the search engine and we acquire potential noun compounds from them. Then, in each snippet, we look for an instantiation of the pattern used in the query and we try to extract suitable noun(s) that occupy the position(s) of the ⋆.

For *loose bootstrapping*, we extract two nouns, one from each end of the matched pattern, while for *strict bootstrapping* and for *NC-only strict bootstrapping*, we only extract one noun, either preceding or following the pattern, since the other noun is already fixed. We then lemmatize the extracted noun(s) and we form NC candidates from the two arguments of the instantiated pattern, taking into account whether the pattern is active or passive.

Due to the vast number of snippets we have to process, we decided not to use a syntactic parser or a part-of-speech (POS) tagger[1]; thus, we use heuristic rules instead. We extract "phrases" using simple indicators such as punctuation (e.g., comma, period), coordinating conjunctions[2] (e.g., *and, or*), prepositions (e.g., *at, of, from*), subordinating conjunctions (e.g., *because, since, although*), and relative pronouns (e.g., *that, which, who*). We then extract the nouns from these phrases, we lemmatize them using WordNet, and we form a list of NC candidates.

While the above heuristics work reasonably well in practice, we perform some further filtering, removing all NC candidates for which one or more of the following conditions are met:

1. the candidate NC is one of the seed examples or has been extracted on a previous iteration;
2. the head and the modifier are the same;
3. the head or the modifier are not both listed as nouns in WordNet (Fellbaum, 1998);
4. the candidate NC occurs less than 100 times in the *Google Web 1T 5-gram corpus*[3];
5. the NC is extracted less than N times (we tried 5 and 10) in the context of the pattern for all instantiations of the pattern.

4.2 Bootstrapping Step 2: Pattern Extraction

This is the second step of our bootstrapping algorithm as shown on Figure 1. Given a list of noun compounds, we mine the Web to extract patterns: verbs and/or prepositions that can paraphrase each NC. The idea is to turn the NC's pre-modifier into a post-modifying relative clause and to collect the verbs and prepositions that are used in such clauses. Below we describe each of the sub-steps of the NC extraction process: query generation, snippet harvesting, and NC extraction & filtering.

4.2.1 Query Generation

The process of extraction starts with exact-phrase queries issued against a Web search engine (again *Yahoo!*) using the following generalized pattern:

"HEAD THAT? ⋆ MOD"

where MOD and HEAD are inflected forms of NC's modifier and head, respectively, THAT? stands for *that*, *which*, *who* or the empty string, and ⋆ stands for 1-6 instances of search engine's star operator.

For example, given *orange juice*, we could generate queries like "juice that ⋆ oranges", "juices which ⋆ ⋆ ⋆ ⋆ ⋆ ⋆ oranges", and "juices ⋆ ⋆ ⋆ orange".

4.2.2 Snippet Extraction

The same as in Section 4.1.2 above.

[1] In fact, POS taggers and parsers are unreliable for Web-derived snippets, which often represent parts of sentences and contain errors in spelling, capitalization and punctuation.

[2] Note that filtering the arguments using such indicators indirectly subsumes the pattern "X PATTERN Y and" proposed in (Kozareva and Hovy, 2010).

[3] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13

4.2.3 Pattern Extraction and Filtering

We split the extracted snippets into sentences, and filter out all incomplete ones and those that do not contain (a possibly inflected version of) the target nouns. We further make sure that the word sequence following the second mentioned target noun is non-empty and contains at least one non-noun, thus ensuring the snippet includes the entire noun phrase. We then perform shallow parsing, and we extract all verb forms, and the following preposition, between the target nouns. We allow for adjectives and participles to fall between the verb and the preposition but not nouns; we further ignore modal verbs and auxiliaries, but we retain the passive *be*, and we make sure there is exactly one verb phrase between the target nouns. Finally, we lemmatize the verbs to form the patterns candidates, and we apply the following pattern selection rules:

1. we filter out all patterns that were provided as initial seeds or were extracted previously;
2. we select the top 20 most frequent patterns;
3. we filter out all patterns that were extracted less than N times (we tried 5 and 10) and with less than M NCs per pattern (we tried 20 and 50).

5 Target Relation and Seed Examples

As we mentioned above, we use the inventory of abstract relations proposed in the popular theoretical linguistics theory of Levi (1978). In this theory, noun compounds are derived from underlying relative clauses or noun phrase complement constructions by means of two general processes: predicate deletion and predicate nominalization. Given a two-argument predicate, *predicate deletion* removes that predicate, but retains its arguments to form an NC, e.g., *pie made of apples* $\rightarrow$ *apple pie*. In contrast, *predicate nominalization* creates an NC whose head is a nominalization of the underlying predicate and whose modifier is either the subject or the object of that predicate, e.g., *The President refused General MacArthur's request.* $\rightarrow$ *presidential refusal*.

According to Levi, predicate deletion can be applied to abstract predicates, whose semantics can be roughly approximated using five paraphrasing verbs (CAUSE, HAVE, MAKE, USE, and BE) and four prepositions (IN, FOR, FROM, and ABOUT).

Typically, in predicate deletion, the modifier is derived from the object of the underlying relative clause; however, the first three verbs also allow for it to be derived from the subject. Levi expresses the distinction using indexes. For example, *music box* is MAKE_1 (object-derived), i.e., the *box makes music*, while *chocolate bar* is MAKE_2 (subject-derived), i.e., *the bar is made of chocolate* (note the passive).

Due to time constraints, we focused on one relation of Levi's, MAKE_2, which is among the most frequent relations an NC can express and is present in some form in many relation inventories (Warren, 1978; Barker and Szpakowicz, 1998; Rosario and Hearst, 2001; Nastase and Szpakowicz, 2003; Girju et al., 2005; Girju et al., 2007; Girju et al., 2009; Hendrickx et al., 2010; Tratz and Hovy, 2010).

In Levi's theory, MAKE_2 means that the head of the noun compound is made up of or is a product of its modifier. There are three subtypes of this relation (we do not attempt to distinguish between them):

(a) the modifier is a unit and the head is a configuration, e.g., *root system*;

(b) the modifier represents a material and the head is a mass or an artefact, e.g., *chocolate bar*;

(c) the head represents human collectives and the modifier specifies their membership, e.g., *worker teams*.

There are 20 instances of MAKE_2 in the appendix of (Levi, 1978), and we use them all as *seed NCs*. As *seed patterns*, we use a subset of the human-proposed paraphrasing verbs and prepositions corresponding to these 20 NCs in the dataset in (Nakov, 2008b), where each NC is paraphrased by 25-30 annotators. For example, for *chocolate bar*, we find the following list of verbs (the number of annotators who proposed each verb is shown in parentheses):

> be made of (16), contain (16), be made from (10), be composed of (7), taste like (7), consist of (5), be (3), have (2), melt into (2), be manufactured from (2), be formed from (2), smell of (2), be flavored with (1), sell (1), taste of (1), be constituted by (1), incorporate (1), serve (1), contain (1), store (1), be made with (1), be solidified from (1), be created from (1), be flavoured with (1), be comprised of (1).

Seed NCs: bronze statue, cable network, candy cigarette, chocolate bar, concrete desert, copper coin, daisy chain, glass eye, immigrant minority, mountain range, paper money, plastic toy, sand dune, steel helmet, stone tool, student committee, sugar cube, warrior castle, water drop, worker team
Seed patterns: be composed of, be comprised of, be inhabited by, be lived in by, be made from, be made of, be made up of, be manufactured from, be printed on, consist of, contain, have, house, include, involve, look like, resemble, taste like

Table 1: Our seed examples: 20 noun compounds and 18 verb patterns.

As we can see, the most frequent patterns are of highest quality, e.g., *be made of (16)*, while the less frequent ones can be wrong, e.g., *serve (1)*. Therefore, we filtered out all verbs that were proposed less than five times with the 20 seed NCs. We further removed the verb *be*, which is too general, thus ending up with 18 seed patterns. Note that some patterns can paraphrase multiple NCs: the total number of seed NC-pattern pairs is 84.

The seed NCs and patterns are shown in Table 1. While some patterns, e.g., *taste like* do not express the target relation MAKE$_2$, we kept them anyway since they were proposed by several human annotators and since they do express the fine-grained semantics of some particular instances of that relation; thus, we thought they might be useful, even for the general relation. For example, *taste like* has been proposed 8 times for *candy cigarette*, 7 times for *chocolate bar*, and 2 times for *sugar cube*, and thus it clearly correlates well with some seed examples, even if it does not express MAKE$_2$ in general.

6 Experiments and Evaluation

Using the NCs and patterns in Table 1 as initial seeds, we ran our algorithm for three iterations of *loose bootstrapping* and *strict bootstrapping*, and for two iterations of *NC-only strict bootstrapping*. We only performed up to three iterations because of the huge number of noun compounds extracted for *NC-only strict bootstrapping* (which we only ran for two iterations) and because of the low number of new NCs extracted by *loose bootstrapping* on iteration 3. While we could have run *strict bootstrapping* for more iterations, we opted for a comparable number of iterations for all three methods.

Examples of noun compounds that we have extracted are *bronze bell* (be made of, be made from) and *child team* (be composed of, include). Example patterns are *be filled with* (cotton bag, water cup) and *use* (water sculpture, wood statue).

Limits (see 4.2.3)	**Extracted & Retained** NCs	Patterns	Patt.+NC
Loose Bootstrapping			
N=5, M=50	1,662 / 61.67	12 / 65.83	1,337
N=10, M=20	590 / 61.52	9 / 65.56	316
Strict Bootstrapping			
N=5, M=50	25,375 / 67.42	16 / 71.43	9,760
N=10, M=20	16,090 / 68.27	16 / 78.98	5,026
NC-only Strict Bootstrapping			
N=5	205,459 / 69.59	–	–
N=10	100,550 / 70.43	–	–

Table 2: Total number and accuracy in % for NCs, patterns and NC-pattern pairs extracted and retained for each of the three methods over all iterations.

Tables 2 and 3 show the overall results. As we mentioned in section 4.2.3, at each iteration, we filtered out all patterns that were extracted less than N times or with less than M NCs. Note that we only used the 10 most frequent NCs per pattern as NC seeds for NC extraction in the next iteration of *strict bootstrapping* and *NC-only strict bootstrapping*. Table 3 shows the results for two value combinations of (N;M): (5;50) and (10;20). Note also that if some NC was extracted by several different patterns, it was only counted once. Patterns are subject to particular NCs, and thus we show (1) the number of patterns extracted with all NCs, i.e., unique NC-pattern pairs, (2) the accuracy of these pairs,[4] and (3) the number of unique patterns retained after filtering, which will be used to extract new noun compounds on the second step of the current iteration.

[4]One of the reviewers suggested that evaluating the accuracy of NC-pattern pairs could potentially conceal some of the drift of our algorithm. For example, while *water cup* / *be filled with* is a correct NC-pattern pair, *water cup* is incorrect for MAKE$_2$; it is probably an instance of Levi's FOR. Thus, the same bootstrapping technique evaluated against a fixed set of semantic relations (which is the more traditional approach) could arguably show bootstrapping going "off the rails" more quickly than what we observe here. However, our goal, as stated in Section 3, is to find NC-specific paraphrases, and our evaluation methodology is more adequate with respect to this goal.

Limits (see 4.2.3)	Seeds Patt.	Seeds NCs	Iteration 1 Patt.	Iteration 1 NCs	Iteration 2 Patterns	Iteration 2 NCs	Iteration 3 Patterns	Iteration 3 NCs
Loose Bootstrapping								
N=5, M=50	–	18	–	1,144 / 63.11	1,136 / 64.44 / 9	390 / 58.72	201 / 70.00 / 3	128 / 57.03
N=10, M=20	–	18	–	502 / 61.55	294 / 62.50 / 8	78 / 60.26	22 / 90.00 / 1	10 / 70.00
Strict Bootstrapping								
N=5, M=50	20	18	–	7,011 / 70.65	5,312 / 74.00 / 10	11,214 / 67.15	4,448 / 60.00 / 6	7,150 / 64.69
N=10, M=20	20	18	–	4,826 / 71.26	2,838 / 79.38 / 10	7,371 / 67.26	2,188 / 78.33 / 6	3,893 / 66.48
NC-only Strict Bootstrapping								
N=5	20	18	–	7,011 / 70.65	–	198,448 / 69.55	–	–
N=10	20	18	–	4,826 / 71.26	–	95,524 / 70.59	–	–

Table 3: *Evaluation results for up to three iterations.* For NCs, we show the number of unique NCs extracted and their accuracy in %. For patterns, we show the number of unique NC-pattern pairs extracted, their accuracy in %, and the number of unique patterns retained and used to extract NCs on the second step of the current iteration. The first column shows the pattern filtering thresholds used (see Section 4.2.3 for details).

The above accuracies were calculated based on human judgments by an experienced, well-trained annotator. We also hired a second annotator for a small subset of the examples.

For NCs, the first annotator judged whether each NC is an instance of MAKE$_2$. All NCs were judged, except for iteration 2 of *NC-only strict bootstrapping*, where their number was prohibitively high and only the most frequent noun compounds extracted for each modifier and for each head were checked: 9,004 NCs for N=5 and 4,262 NCs for N=10.

For patterns, our first annotator judged the correctness of the unique NC-pattern pairs, i.e., whether the NC is paraphrasable with the target pattern. Given the large number of NC-pattern pairs, the annotator only judged patterns with their top 10 most frequent NCs. For example, if there were 5 patterns extracted, then the NC-pattern pairs to be judged would be no more than $5 \times 10 = 50$.

Our second annotator judged 340 random examples: 100 NCs and 20 patterns with their top 10 NCs for each iteration. The Cohen's kappa (Cohen, 1960) between the two annotators is .66 (85% initial agreement), which corresponds to substantial agreement (Landis and Koch, 1977).

7 Discussion

Tables 2 and 3 show that fixing one of the two nouns in the pattern, as in *strict bootstrapping* and *NC-only strict bootstrapping*, yields significantly higher accuracy (χ^2 test) for both NC and NC-pattern pair extraction compared to *loose bootstrapping*.

The accuracy for *NC-only strict bootstrapping* is a bit higher than for *strict bootstrapping*, but the actual differences are probably smaller since the evaluation of the former on iteration 2 was done for the most frequent NCs, which are more accurate.

Note that the number of extracted NCs is much higher with the strict methods because of the higher number of possible instantiations of the generalized query patterns. For *NC-only strict bootstrapping*, the number of extracted NCs grows exponentially since the number of patterns does not diminish as in the other two methods. The number of extracted patterns is similar for the different methods since we select no more than 20 of them per iteration.

Overall, the accuracy for all methods decreases from one iteration to the next since errors accumulate; still, the degradation is slow. Note also the exception of *loose bootstrapping* on iteration 3.

Comparing the results for N=5 and N=10, we can see that, for all three methods, using the latter yields a sizable drop in the number of extracted NCs and NC-pattern pairs; it also tends to yield a slightly improved accuracy. Note, however, the exception of *loose bootstrapping* for the first two iterations, where the less restrictive N=5 is more accurate.

As a comparison, we implemented the method of Kim and Baldwin (2007), which generates new semantically interpreted NCs by replacing either the head or the modifier of a seed NC with suitable synonyms, hypernyms and sister words from WordNet, followed by similarity filtering using `WordNet::Similarity` (Pedersen et al., 2004).

Rep.	Iter. 1	Iter. 2	Iter. 3	All
Syn.	11/81.81	3/66.67	0	14/78.57
Hyp.	27/85.19	35/77.14	33/66.67	95/75.79
Sis.	381/82.05	1,736/69.33	17/52.94	2,134/75.12
All	419/82.58	1,774/71.68	50/62.00	2,243/**75.47**

Table 4: Number of extracted noun compounds and accuracy in % for the method of Kim and Baldwin (2007). The abbreviations *Syn.*, *Hyp.*, and *Sis.* indicate using synonyms, hypernyms, and sister words, respectively.

The results for three bootstrapping iterations using the same list of 20 initial seed NCs as in our previous experiments, are shown in Table 4. We can see that the overall accuracy of their method is slightly better than ours. Note, however, that our method acquired a much larger number of NCs, while allowing more variety in the NC semantics. Moreover, for each extracted noun compound, we also generated a list of fine-grained paraphrasing verbs.

8 Error Analysis

Below we analyze the errors of our method.

Many problems were due to wrong POS assignment. For example, on Step 2, because of the omission of *that* in "*the statue has such high quality gold (that) demand is ...*", *demand* was tagged as a noun and thus extracted as an NC modifier instead of *gold*. The problem also arose on Step 1, where we used WordNet to check whether the NC candidates were composed of two nouns. Since words like *clear*, *friendly*, and *single* are listed in WordNet as nouns (which is possible in some contexts), we extracted wrong NCs such as *clear cube*, *friendly team*, and *single chain*. There were similar issues with verb-particle constructions since some particles can be used as nouns as well, e.g., *give back, break down*.

Some errors were due to semantic transparency issues, where the syntactic and the semantic head of a target NP were mismatched (Fillmore et al., 2002; Fontenelle, 1999). For example, from the sentence "*This wine is made from a range of white grapes.*", we would extract *range* rather than *grapes* as the potential modifier of *wine*.

In some cases, the NC-pattern pair was correct, but the NC did not express the target relation, e.g., while *contain* is a good paraphrase for *toy box*, the noun compound itself is not an instance of MAKE$_2$.

There were also cases where the pair of extracted nouns did not make a good NC, e.g., *worker work* or *year toy*. Note that this is despite our checking that the candidate NC occurred at least 100 times in the *Google Web 1T 5-gram corpus* (see Section 4.1.3). We hypothesized that such bad NCs would tend to have a low collocation strength. We tested this hypothesis using the Dice coefficient, calculated using the *Google Web 1T 5-gram corpus*. Figure 2 shows a plot of the NC accuracy vs. collocation strength for *strict bootstrapping* with N=5, M=50 for all three iterations (the results for the other experiments show a similar trend). We can see that the accuracy improves slightly as the collocation strength increases: compare the left and the right ends of the graph (the results are mixed in the middle though).

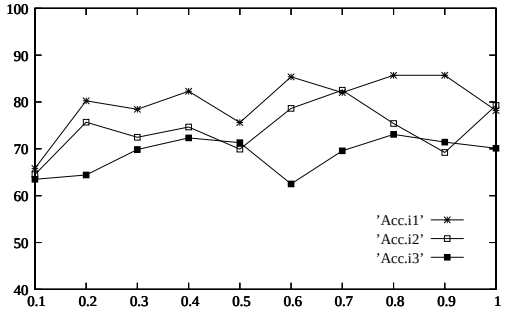


Figure 2: NC accuracy vs. collocation strength.

9 Conclusion and Future Work

We have presented a framework for building a very large dataset of noun compounds expressing a given target abstract semantic relation. For each extracted noun compound, we generated a corresponding fine-grained semantic interpretation: a frequency distribution over suitable paraphrasing verbs.

In future work, we plan to apply our framework to the remaining relations in the inventory of Levi (1978), and to release the resulting dataset to the research community. We believe that having a large-scale dataset of noun compounds interpreted with both fine- and coarse-grained semantic relations would be an important contribution to the debate about which representation is preferable for different tasks. It should also help the overall advancement of the field of noun compound interpretation.

Acknowledgments

This research is partially supported (for the second author) by the *SmartBook project*, funded by the Bulgarian National Science Fund under Grant D002-111/15.12.2008.

We would like to thank the anonymous reviewers for their detailed and constructive comments, which have helped us improve the paper.

References

Ken Barker and Stan Szpakowicz. 1998. Semi-automatic recognition of noun modifier relationships. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 96–102.

Matthew Berland and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, ACL '99, pages 57–64.

Cristina Butnariu and Tony Veale. 2008. A concept-centered approach to noun-compound interpretation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, COLING '08, pages 81–88.

Cristina Butnariu, Su Nam Kim, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2009. Semeval-2010 task 9: The interpretation of noun compounds using paraphrasing verbs and prepositions. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, SEW '09, pages 100–105.

Cristina Butnariu, Su Nam Kim, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2010. SemEval-2010 task 9: The interpretation of noun compounds using paraphrasing verbs and prepositions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval-2, pages 39–44.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 1(20):37–46.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics*, PACLING '07, pages 172–180.

Pamela Downing. 1977. On the creation and use of English compound nouns. *Language*, 53:810–842.

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

Charles J. Fillmore, Collin F. Baker, and Hiroaki Sato. 2002. Seeing arguments through transparent structures. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, volume III of *LREC '02*, pages 787–791.

Timothy Wilking Finin. 1980. *The semantic interpretation of compound nominals*. Ph.D. thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA. AAI8026491.

Thierry Fontenelle. 1999. Semantic resources for word sense disambiguation: a sine qua non. *Linguistica e Filologia*, 9:25–41.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19(44):479–496.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the 4th Semantic Evaluation Workshop*, SemEval-1, pages 13–18.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2009. Classification of semantic relations between nominals. *Language Resources and Evaluation*, 43(2):105–121.

Roxana Girju. 2007. Improving the interpretation of noun phrases with cross-linguistic information. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL '07, pages 568–575.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, COLING '92, pages 539–545.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval-2, pages 33–38.

Su Nam Kim and Timothy Baldwin. 2005. Automatic interpretation of compound nouns using WordNet similarity. In *Proceedings of 2nd International Joint Conference on Natural Language Processing*, IJCNLP '05, pages 945–956.

Su Nam Kim and Timothy Baldwin. 2006. Interpreting semantic relations in noun compounds via verb semantics. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics and 21st International Conference on Computational Linguistics*, ACL-COLING '06, pages 491–498.

Su Nam Kim and Timothy Baldwin. 2007. Interpreting noun compounds using bootstrapping and sense collocation. In *Proceedings of Conference of the Pacific Association for Computational Linguistics*, PACLING '07, pages 129–136.

Zornitsa Kozareva and Eduard Hovy. 2010. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1482–1491.

Richard J. Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Maria Lapata. 2002. The disambiguation of nominalizations. *Computational Linguistics*, 28(3):357–388.

Mark Lauer. 1995. *Designing Statistical Language Learners: Experiments on Noun Compounds*. Ph.D. thesis, Dept. of Computing, Macquarie University, Australia.

Judith Levi. 1978. *The Syntax and Semantics of Complex Nominals*. Academic Press, New York, USA.

Tara McIntosh and James Curran. 2009. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL-IJCNLP '09, pages 396–404.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the semantic classification of noun phrases. In *Proceedings of the HLT-NAACL'04 Workshop on Computational Lexical Semantics*, pages 60–67.

Preslav Nakov and Marti A. Hearst. 2006. Using verbs to characterize noun-noun relations. In *Proceedings of the 12th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, AIMSA '06, pages 233–244.

Preslav Nakov and Marti Hearst. 2008. Solving relational similarity problems using the web as a corpus. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, ACL '08, pages 452–460.

Preslav Nakov. 2008a. Improved Statistical Machine Translation Using Monolingual Paraphrases. In *Proceedings of the 18th European Conference on Artificial Intelligence*, ECAI '08, pages 338–342.

Preslav Nakov. 2008b. Noun compound interpretation using paraphrasing verbs: Feasibility study. In *Proceedings of the 13th international conference on Artificial Intelligence: Methodology, Systems, and Applications*, AIMSA '08, pages 103–117.

Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Proceedings of the 5th International Workshop on Computational Semantics*, pages 285–301.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, AAAI '04, pages 1024–1025.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, AAAI '99, pages 474–479.

Barbara Rosario and Marti Hearst. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing*, EMNLP '01, pages 82–90.

Barbara Rosario and Marti Hearst. 2002. The descent of hierarchy, and selection in relational semantics. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, ACL '02, pages 247–254.

Diarmuid Ó Séaghdha. 2009. Semantic classification with WordNet kernels. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL '09, pages 237–240.

Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In *Proceedings of 16th International World Wide Web Conference*, WWW '07, pages 697–706.

Marta Tatu and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, HLT-EMNLP '05, pages 371–378.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, EMNLP '02, pages 214–221.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 678–687.

Lucy Vanderwende. 1994. Algorithm for automatic interpretation of noun sequences. In *Proceedings of the 15th Conference on Computational linguistics*, pages 782–788.

Beatrice Warren. 1978. Semantic patterns of noun-noun compounds. In *Gothenburg Studies in English 41, Goteburg, Acta Universtatis Gothoburgensis*.

Linguistic Redundancy in Twitter

Fabio Massimo Zanzotto
University of Rome "Tor Vergata"
Rome, Italy
zanzotto@info.uniroma2.it

Marco Pennacchiotti
Yahoo! Labs
Sunnyvale, CA, 94089
pennac@yahoo-inc.com

Kostas Tsioutsiouliklis
Yahoo! Labs
Sunnyvale, CA, 94089
kostas@yahoo-inc.com

Abstract

In the last few years, the interest of the research community in micro-blogs and social media services, such as Twitter, is growing exponentially. Yet, so far not much attention has been paid on a key characteristic of micro-blogs: the high level of information redundancy. The aim of this paper is to systematically approach this problem by providing an operational definition of redundancy. We cast redundancy in the framework of Textual Entailment Recognition. We also provide quantitative evidence on the pervasiveness of redundancy in Twitter, and describe a dataset of redundancy-annotated tweets. Finally, we present a general purpose system for identifying redundant tweets. An extensive quantitative evaluation shows that our system successfully solves the redundancy detection task, improving over baseline systems with statistical significance.

1 Introduction

Micro-blogs and social media services, such as Twitter, have experienced an exponential growth in the last few years. The interest of the research community and the industry in these services has followed a similar trend. Web companies such as Google, Yahoo, and Bing are integrating more and more social content to their sites. At the same time, the computational linguistic community is getting increasingly interested in studying social and linguistic properties of Twitter and other micro-blogs (Java et al., 2007; Krishnamurthy et al., 2008; Kwak et al., 2010; Zhao et al., 2007; Popescu and Pennacchiotti, 2010; Petrović et al., 2010; Lin et al., 2010; Liu et al., 2010; Ritter et al., 2010). Yet, so far, not much attention has been paid on a key characteristic of micro-blogs: the high level of information redundancy. Users often post messages with the same, or very similar, content, especially when reporting or commenting on news and events. For example, the following two tweets are part of a large set of redundant tweets issued during the 2010 winter Olympics:

(example 1)

t_1 : *"Swiss ski jumper Simon Ammann takes first gold of Vancouver"*

t_2 : *"Swiss (Suisse) get the Gold on Normal Hill ski jump. #Vancouver2010"*

By performing an editorial study (described later in the paper) we discovered that a large part of event-related tweets are indeed redundant.

Detecting information redundancy is important for various reasons. First, most applications based on Twitter share the goal of providing tweets that are both *informative* and *diverse*, with respect to an initial user information need. For example, Twitter search engines should ideally select the most informative and diverse set of tweets in return to a user query. Similarly, a news web portal that attaches tweets to a given news article should attach those tweets that provide the broadest and most diverse set of information, opinions, and updates about the news item. To keep a high level of diversity, redundant tweets should be removed from the set of tweets displayed to the user. Figure 1 shows an example of a Twitter search engine where redundant tweets are

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 659–669,
Edinburgh, Scotland, UK, July 27–31, 2011.

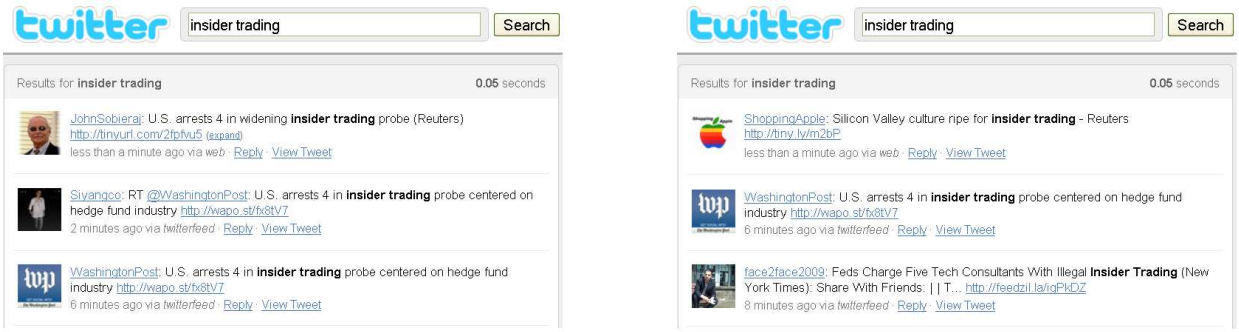


Figure 1: Twitter search: actual Twitter results and desired results after redundancy reduction.

present (left) and where they are discarded (right).

Also, from a computational linguistic point of view, the high redundancy in micro-blogs gives the unprecedented opportunity to study classical tasks such as text summarization (Haghighi and Vanderwende, 2009), textual entailment recognition (Dagan et al., 2006) and paraphrase detection (Dolan et al., 2004) on very large corpora characterized by an original and emerging linguistic style, pervaded with ungrammatical and colloquial expressions, abbreviations, and new linguistic forms.

The aim of this paper is to formally define, for the first time, the problem of redundancy in micro-blogs and to systematically approach the task of automatic redundancy detection. Note that we focus on linguistic redundancy, i.e. tweets that convey the same information with different wordings, and ignore the more trivial issue of detecting retweets, which can be considered the most basic expression of redundancy.

The main contributions of this paper are the following:

- We formally define the problem of redundancy detection in micro-blogs within the framework of Textual Entailment theory;
- We report results from an editorial study and provide quantitative evidence of the pervasiveness of redundancy in Twitter;
- We present a set of simple and effective machine learning models for solving the task of redundancy detection;
- We provide promising experimental results that show that these models outperform baseline approaches with statistical significance, and we report a qualitative evaluation revealing the advantages of the proposed model.

The rest of the paper is organized as follows. First, we shortly describe related work in Section 2. Next, we provide our operational definition of redundancy and introduce our editorial study and dataset in Section 3. In Section 4 we describe our models for redundancy detection. In Section 5 we provide a quantitative and qualitative evaluation of our models. In Section 6 we conclude the paper with final observations and future work.

2 Related Work

So far, most research on **Twitter** has focused on its network structure, the social behavior of its users (Java et al., 2007; Krishnamurthy et al., 2008; Kwak et al., 2010), ranking tweets by relevance for web search (Ramage et al., 2010; Duan et al., 2010), and the analysis of time series for extracting trending news, events and facts (Zhao et al., 2007; Popescu and Pennacchiotti, 2010; Petrović et al., 2010; Lin et al., 2010). Only few studies have specifically focused on the linguistic content analysis of tweets, e.g. (Davidov et al., 2010; Barbosa and Feng, 2010). To date, our paper most closely relates to works on semantic role labeling (SRL) on social media (Liu et al., 2010) and conversation modeling (Ritter et al., 2010).

Liu et al. (2010) present a self-learning SRL system for news tweets, with the goal of addressing low performance caused by the noise and the unstructured nature of the data. The authors first cluster together tweets that refer to the same news. Then, for each cluster, they identify the tweets that are

well-formed (i.e. copy-pasted from news), and induce role mappings between well-formed and noisy tweets in the same cluster by performing word alignment. In our paper we are also interested in aligning and grouping tweets, although our goal is to detect redundancy, not to perform SRL.

On a different ground, Ritter et al. (2010) propose a probabilistic model to discover dialogue acts in Twitter conversations and to classify tweets in a conversation according to those acts. (A conversation is defined as a set of tweets in the same reply thread.) The authors define 10 major dialogue acts for Twitter, including status, question, response and reaction, and automatically build a probabilistic transition graph for such acts. In our paper, we also aim at classifying tweets, but our interest is in information redundancy instead of acts.

In the computational linguistic literature, **redundancy detection** is studied in multi-document summarization, where the overall document is used to select the most informative sentences or snippets (Haghighi and Vanderwende, 2009). Since tweets are short and tweet sets cannot be considered documents, these methods are hard to apply. A more convenient setting is paraphrase detection (Dolan et al., 2004) and textual entailment recognition (Dagan et al., 2006) (RTE).

In RTE the task is to recognize if a text called the *text* T (typically one or two sentences long) entails another text called the *hypothesis* H. Many approaches have been proposed for this task, mostly based on machine learning. Three main classes of features have been so far explored in RTE: distance/similarity feature spaces (Corley and Mihalcea, 2005; Newman et al., 2005; Haghighi et al., 2005; Hickl et al., 2006), entailment trigger feature spaces (de Marneffe et al., 2006; MacCartney et al., 2006), and pair content feature spaces (Zanzotto et al., 2009). Distance/similarity feature spaces are more suitable to the paraphrase detection task because they model the similarity between the two texts. On the other hand, entailment trigger and content feature spaces model complex relations between the texts, taking into account first-order entailment rules, i.e. entailment rules with variables.

In this paper, one of our goals is to explore RTE techniques and features that are usually used for classical texts, and check if they can be successfully adapted to the unstructured, and oftentimes ungrammatical, Twitter language.

3 Redundancy in Twitter

We formally define two tweets as **redundant** if they either convey the same information (*paraphrase*) or if the information of one tweet subsumes the information of the other (*textual entailment*). For example, the pair in *(example 1)* is redundant. The first tweet subsumes (i.e. 'textually entails') the other; both tweets state that Switzerland won a Gold Medal at the Vancouver winter Olympics, but the first one also specifies the name of the athlete. The following pair is, instead, non-redundant, because the two tweets convey different information, and they do not subsume each other:

(example 2)

t_1 : *"Goal! Iniesta scores for #ESP and they have one hand on the #worldcup"*

t_2 : *"this will be a hard final #Esp vs Ned #worldcup"*

Our definition of redundancy is grounded on, and inspired by, the theory of Textual Entailment, to which we refer the reader for further details (Dagan et al., 2006).

3.1 Quantifying redundancy

How pervasive is redundancy in Twitter? In order to answer this question we performed an initial editorial study where human editors were asked to annotate pairs of tweets as being either redundant or non-redundant. The editorial study also serves as a test bed for evaluating our redundancy detection models, as discussed in Section 5.

In the study we focus on 'informative' tweets, i.e. tweets that describe or comment on relevant events/facts. Indeed, these are the types of tweets for which redundancy is a critical issue, especially in view of real applications, e.g. to present a diverse set of tweets for a given news article. Other types of tweets, such as status updates, self-promotions, and personal messages are of less interest in this context.

Dataset extraction. The study is performed on an automatically built dataset of informative tweets. The most critical issue for extracting the dataset is to pre-process tweets and to discard those that are

not informative. This is not an easy task: a recent study (Pear-Analytics, 2009) estimates that only 4% of all tweets are factual news, and only 37% are conversations with content. The rest are spam, status updates and other types of uninformative content. In order to retain only informative tweets we first extract *buzzy snapshots* (Popescu and Pennacchiotti, 2010). A snapshot is defined as a set of tweets that explicitly mention a specific topic within a specified time period. A buzzy snapshot is defined as a snapshot with a large number of tweets, compared to previous time periods. For example, given the topic 'Haiti earthquake', the snapshot composed by the tweets mentioning 'Haiti earthquake' on January 12th, 2010, will constitute a buzzy snapshot, since in previous days the topic was not mentioned often.

We use two different topic lists: a *celebrity list* containing about 104K celebrity names, crawled from Wikipedia, including actors, musicians, politicians, and athletes; and an *event list* composed of 398 hashtags related to 8 major events that happened between January and July 2010, and listed in Wikipedia: [1] the earthquake in Haiti, the winter Olympics, the earthquake in Chile, the death of the Polish president, the volcano eruption in Iceland, the oil spill in the Gulf of Mexico, the Greek financial crisis, and the FIFA world cup.

We extract buzzy snapshots for the above two topic lists by following the method described in (Popescu and Pennacchiotti, 2010): we consider time periods of one day, and call buzzy the snapshots that mention a given topic α times more than the average over the previous 2 days. We set α to 20 and 5 respectively for the celebrity list and the event list. We further exclude irrelevant and spam snapshots by removing those that have: fewer than 10 tweets; more than 50% of tweets non-English; and an average token overlap between tweets of more than 80%, usually corresponding to spam threads.

The extraction is performed on a Twitter corpus containing all tweets posted between July 2009 and August 2010. In all, we extract 972 snapshots for the celebrity list, containing 205,885 tweets (i.e. average of 212 tweets per snapshot); and 674 snapshots for the event list, containing 393,965 tweets (584 tweets per snapshot).

[1] Hashtags are keywords prefixed by '#', that are used by the Twitter community to mark the topic of a tweet. We collected our set of hashtags by semi-automatically inspecting the Twitter stream in the days the major events happened.

redundant	**367**	**(29.5%)**
entailment	195	(15.7%)
paraphrase	172	(13.5%)
non-redundant	**875**	**(70.5%)**
related	541	(43.6%)
unrelated	334	(26.9%)

Table 1: Results of the redundancy editorial study.

The above two final snapshot corpora (i.e. the 972 celebrities' snapshots and 674 events' snapshots) can be considered a good representation of event descriptions and comments on Twitter, thus forming our initial set of 'informative' tweets. From these two corpora, we extract the final tweet-pair dataset by randomly sampling 1500 pairs of tweets contained in the same snapshot. Tweet-pairs that contain retweets are excluded.

Dataset annotation. The main editorial task consisted of annotating tweet-pairs as either redundant or non-redundant. We also asked editors to characterize the specific linguistic relation between the two tweets of a pair. We consider four relations: *entailment* (the first tweet entails the second or vice versa), *paraphrase*, *contradiction* (the tweets contradict each other), and *related* (the tweets are about the same topic, e.g. the Haiti earthquake, but are in none of the previous relations). Tweets that were about different topics were labeled *unrelated*. Annotators were asked to base their decisions on the parts of the tweets that contained information relevant to the selected topic, e.g. the earthquake in Haiti. These parts were marked in the corpus. Focusing on these parts is in line with potential applications of tweet redundancy detection as tweets are firstly grouped around a topic. Note that pairs that fall under the entailment or paraphrase relation are redundant, while unrelated, related, and contradictory tweets are non-redundant.

The annotation was performed in a three stage process, since tweets are sometimes hard to understand and hence to annotate (misspellings, usage of slang and abbreviations, lack of discourse context). In the first step, the 1500 pairs were independently annotated by a pool of 20 trained editors, super-

vised by an expert lead. In the second step, the annotations were checked by three highly trained experts with background in computational linguistics: each pair was independently checked by two experts. Average kappa agreement in this second step is $kappa = 0.63$ (corresponding to 'good agreement'). In a final step, discordances between the two experts were resolved by the third expert. Unclear and unresolved pairs after the three stages were discarded from the dataset, leaving a final set of 1242 pairs. [2]

Annotation Results. Table 1 reports the results of our study. Among the 1242 tweet-pairs, 367 (30%) are redundant and 875 (70%) are non-redundant. This shows that redundancy is indeed a pervasive phenomenon in Twitter, and a critical issue that has to be solved in order to provide clean and diverse social content. Most cases of redundancy correspond to tweets that report the same fact using different wording, occasionally adding irrelevant personal comments and sentiments (e.g. 'Johnny Depp died' vs. 'OMG, I am so sad that Johnny Depp is dead').

4 Redundancy detection models

The task of **redundancy detection** in Twitter is a tweet-pair classification problem. Given two tweets t_1 and t_2, the goal is to classify the pair (t_1, t_2) as being either redundant or non-redundant.

In this section we describe different models for redundancy detection, inspired by existing work in RTE. We adopt a machine learning approach where a Support Vector Machine (SVM) is trained on a manually annotated training set to classify incoming test examples as either redundant or non-redundant. An evaluation of the different models adopting for training and testing the dataset described in Section 3, is presented in Section 5.

4.1 Bag-of-word model (BOW)

The bag-of-word model is the most simple approach for detecting redundancy. It is used as a **baseline** in our experiment. The simple intuition of the model is that if two tweets t_1 and t_2 have a high lexical overlap, then they are likely to express the same information – i.e. they are likely to be redundant. In this model, the SVM is trained using a single feature that computes the cosine similarity between the bag-of-word vectors of the two tweets. The bag-of-word vector is built using a classical *tf*idf* weighting schema over the set of tokens of the pair. This a very simple baseline as SVM is only learning thresholds using this single feature.

The bag-of-word model is of course a naive approach, since in many cases redundant tweets can have very different lexical content (e.g. the following two tweets: "Farrah Fawcett left out of Oscar memorial", "No Farrah Fawcett's memory at the Academy Awards"), and non-redundant tweets can have similar lexical content (e.g. the tweets: "Johnny Deep is dead", "Johnny Deep is not dead").

4.2 WordNet-based bag-of-word model (WBOW)

The second **baseline** model was first defined in (Corley and Mihalcea, 2005) and since then has been used by many RTE systems. The model extends BOW by measuring similarity at the semantic level, instead of the lexical level.

For example, consider the tweet pair: "Oscars forgot Farrah Fawcett", "Farrah Fawcett snubbed at Academy Awards". This pair is redundant, and, hence, should be assigned a very high similarity. Yet, BOW would assign a low score, since many words are not shared across the two tweets. WBOW fixes this problem by matching 'Oscar'-'Academy Awards' and 'forgot'-'snubbed' at the semantic level. To provide these matches, WBOW relies on specific word similarity measures over WordNet (Miller, 1995), that allow synonymy and hyperonymy matches: in our experiments we specifically use Jiang&Conrath similarity (Jiang and Conrath, 1997).

In practice, we implement WBOW by using the text similarity measure defined in (Corley and Mihalcea, 2005) as the single feature in the SVM classifier that, as in BOW, learns the threshold on this single feature.

4.3 Lexical content model (LEX)

This model and the next ones (SYNT and FOR) explicitly model the content of a tweet pair $P =$

[2] At this time, the Twitter^{TM} Terms of Use do not allow publication of the annotated dataset. Should the Terms of Use change, the dataset will become available for download at *http://art.uniroma2.it/zanzotto/datasets*.

(t_1, t_2) as a whole. This is a radically different approach with respect to the similarity-based models explored so far, where the content of t_1 and t_2 were treated independently (i.e. each tweet with its own bag of words), and the SVM used as the single feature the similarity between the two tweets.

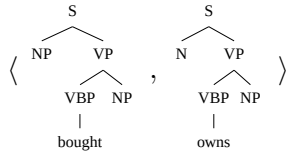
In the LEX model we represent the content of the tweet pair in a double bag-of-word vector space. Each pair $P = (t_1, t_2)$ is represented by two bag-of-word vectors, $(\vec{t_1}, \vec{t_2})$. Within this space, we can then define a specific similarity measure between pairs using a kernel function in the SVM learning algorithm. Given two pairs of tweets $P^{(a)}$ and $P^{(b)}$, the LEX kernel function is defined as follows:

$$K_{LEX}(P^{(a)}, P^{(b)}) = cos(t_1^{(a)}, t_1^{(b)}) + cos(t_2^{(a)}, t_2^{(b)})$$

where $cos(\cdot, \cdot)$ is the cosine similarity between the two vectors. The LEX feature space is simple and can be extremely effective in modeling the content of tweet pairs. Yet, in principle, it doesn't model the relations among words in the tweet. Different content feature spaces are then needed to capture these relations.

4.4 Syntactic content model (SYNT)

The SYNT model represents a tweet pair using pairs of syntactic tree fragments from t_1 and t_2. Each feature is a pair $< fr_1, fr_2 >$, where fr_1 and fr_2 are syntactic tree fragments (see figure below). As defined in (Collins and Duffy, 2002), a syntactic tree fragment fr_i is active in t_i when fr_i is a subtree of the syntactic interpretation of t_i. Therefore, these features represent ground rules connecting the left-hand sides and the right-hand sides of the tweet pair: each feature is active for a pair (t_1, t_2) when the left-hand side fr_1 is activated by the syntactic analysis of t_1 and the right-hand side fr_2 is activated by t_2. As an example consider the feature:



This feature is active for the pair of tweets ("*GM bought Opel*","*GM owns Opel*") since the syntactic analysis of the pair matches the feature (given that the two tweets are correctly syntactically analyzed). This feature space models the relations between words syntactically. Therefore it overcomes the limitations of the LEX feature space. But it also introduces a new limitation: the above feature is in fact also active for the tweet pair ("*GM bought Opel*","*Opel owns GM*"). This pair is extremely different from the previous one, thus possibly misleading the classifier.

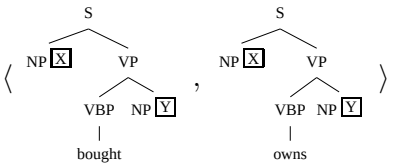
This feature space is not represented explicitly, but it is encoded in a kernel function. Given two pairs of tweets $P^{(a)}$ and $P^{(b)}$, the SYNT kernel function is defined as follows:

$$K_{SYNT}(P^{(a)}, P^{(b)}) = K(t_1^{(a)}, t_1^{(b)}) + K(t_2^{(a)}, t_2^{(b)})$$

where $K(\cdot, \cdot)$ is the tree kernel function described in (Collins and Duffy, 2002).

4.5 Syntactic first-order rule content model (FOR)

The FOR model overcomes the limitations of SYNT, by enriching the space with features representing first-order relations between the two tweets of a pair. Each feature represents a rule with variables, i.e. a first order rule that is activated by the tweet pairs if the variables are unified. This feature space has been introduced in (Zanzotto and Moschitti, 2006) and shown to improve over the ones above. Each feature $< fr_1, fr_2 >$ is a pair of syntactic tree fragments augmented with variables. The feature is active for a tweet pair (t_1, t_2) if the syntactic interpretations of t_1 and t_2 can be unified with $< fr_1, fr_2 >$. For example, consider the following feature:



This feature is active for the pair ("*GM bought Opel*","*GM owns Opel*"), with the variable unification X = "*GM*" and Y = "*Opel*". On the contrary, this feature is not active for the pair ("*GM bought Opel*","*Opel owns GM*") as there is no possibility of unifying the two variables. Efficient algorithms for the computation of the related kernel functions can

be found in (Moschitti and Zanzotto, 2007; Zanzotto and Dell'Arciprete, 2009).

5 Experimental Evaluation

In this section we present an evaluation of the different redundancy detection models. First, we define the experimental setup in Section 5.1. Then, we analyze the results of the experiments in Section 5.2.

5.1 Experimental Setup

We experiment with the redundancy detection dataset described in Section 3. We randomly divide the corpus into two sets: 50% for training and 50% for testing. The training set contains 185 positive tweet-pairs and 416 negative pairs. The test set contains 182 positive pairs and 466 negatives.

We evaluate the performance of the SVM models using the following feature combinations: LEX+BOW, LEX+WBOW, SYNT+BOW, SYNT+WBOW, FOR+BOW, FOR+WBOW. We compare to the system baselines BOW and WBOW. [3]

The performance of the different models is computed using the Area Under the ROC curve (AROC) applied to the classification score returned by the SVM. The ROC curve allows us to study the behavior of the classifier in detail, and also provides a powerful way to compare among systems when the dataset is unbalanced (as in our case).

To determine the statistical significance of the difference in the performance of the systems we analyzed, we use the model described in (Yeh, 2000) as implemented in (Padó, 2006).

We pre-process the dataset with the following tools: the Charniak Parser (Charniak, 2000) for parsing sentences, the WordNet similarity package (Pedersen et al., 2004) for computing WBOW and for linking the two tweets in a pair, and SVM-light (Joachims, 1999), extended with the syntactic first-order rule kernels described in (Moschitti and Zanzotto, 2007) for creating the SYNT and the FOR feature spaces. We used the Charniak syntactic parser without any specific adaptation to the Twitter language.

Model	AROC
BOW	0.592
WBOW	0.578
LEX + BOW	0.725 †
LEX + WBOW	0.728 †
SYNT + BOW	0.736 †
SYNT + WBOW	0.737 †
FOR + BOW	0.739 †
FOR + WBOW	**0.747** † ‡

Table 2: Experimental results of the different systems. † indicates statistical significance ($p < 0.01$) with respect to the two baseline methods BOW and WBOW. ‡ indicates statistical significance ($p < 0.1$) with respect to FOR + BOW

5.2 Experimental Results

Table 2 reports the results of the experiment. The first and most important result is that models using content features (LEX, SYNT, and FOR) along with similarity features (BOW and WBOW) outperform the two baseline models using only similarity features with statistical significance, up to more than 15% AROC points.

At first glance, WordNet similarities are not useful: the performance of the WBOW model is indeed comparable and statistically insignificant with respect to the pure token based model BOW. This seems to be intuitive as the language of the tweets can be far from proper English, i.e. it may contain many out-of-dictionary words that are not present in WordNet, thus impairing the similarity measure used by WBOW.

This trend is also confirmed in the case of content-based systems like LEX and SYNT. Using BOW or WBOW in combination with these features has the same effect on the final performance. Only the FOR features are positively affected by the WordNet-based distance. This may be explained by the fact that in the FOR+WBOW system, the WordNet similarity is also used to link words in the two tweets of a pair. This increases the possibility of finding reasonable and useful first-order rules. In the quali-

[3] Note that other feature combinations would not add value, as BOW and WBOW are interchangeable, and the same stands for LEX, SYNT and FOR.

tative analysis that follows, we show some examples that support this intuition.

On the other hand, syntax plays a key role for detecting redundancy. The two syntax based models SYNT and FOR outperform the lexical based models LEX between 1 and 2 AROC points. This is surprising, since the Charniak parser used in the experiments has not been adapted to the Tweet language, and therefore could have produced many interpretation errors, thus impairing the use of syntax. This seems to suggest that if the interpretations of the part-of speech tags of the unknown words is correct, the syntax of tweets is reasonably similar to the syntax of the generic English language.

The best performing model is FOR+WBOW: first-order rules successfully emerge in tweets and are positively exploited by the learning system. In the next section we report examples that support this observation.

5.3 Qualitative analysis

The experimental results reported in the previous section show that first-order syntactic rules in combination with the WordNet-based bag-of-word (FOR+WBOW) are highly effective in detecting redundancy. In this section, we briefly analyze some tweet pairs where the differences between this model and the BOW and WBOW models are evident.

Table 3 reports examples of tweet pairs, along with their ranking position in the test set, according to the SVM score, with respect to different models. The first column represents the editorial gold standard (gs) for the tweet pairs we considered: either redundant (R) or non-redundant (N). Since we feed the classifiers with 'redundant' as the positive class [4], a classifier is better than another if it ranks redundant tweet pairs (R) higher than non-redundant ones (N). The second, the third, and the fourth columns represent the rank given by WBOW+FOR, WBOW, and BOW respectively. The fifth column is the tweet-pair identifier in our dataset (id). The last two columns are the two tweets in each pair.

The table reports interesting examples where redundant pairs have very little lexical similarity while the non-redundant pairs have a high lexical similarity. These are all examples where BOW and WBOW should typically fail, while FOR+WBOW could capture important syntactic first-order rules to overcome the limitations of the pure similarity-based models.

As a first example, both BOW and WBOW fail to assign a high rank (i.e. low rank number) to the redundant pair *o165*: in fact, 'died' does not lexically match 'rip', nor are these two words related in WordNet. In contrast, FOR+WBOW assigns a high rank to this pair, since it may be able to apply the rule $<$*X died*, *rip X*$>$ that was most probably acquired from examples in the training set (the hoax of somebody's death is pervasive in Twitter, and it is therefore likely to fire the abovementioned rule in our dataset if enough examples are available).

The third and the fourth pairs (*o130* and *o21*) show some commonalities [5] . According to the WordNet similarity measure we used, 'recognize' and 'snub' are highly related as well as 'forget' and 'snub'. Hence, the two tokens are linked as similar. For *o130*, the triggering syntactic rule is $<$(S (NP *X*) (VP *Y*),(VP (V *Y*) (NP *X*)$>$ where *X* and *Y* are variables. For *o21*, the rule is: $<$(VP (V *X*) (NP *Y*),(VP (V *X*) (NP *Y*)$>$.

For the non-redundant pairs (N) at the bottom of the table, the first-order rules are less intuitive. Yet, it is clear why these pairs have high lexical similarity (and therefore are ranked high by BOW and WBOW): The two tweets in the pair *oe387* share 'volcanic', 'ash', and the hashtag '#ashtag'. Tweets in *oe64* share 'Icelandic' and 'eruption' but they are describing different facts. Tweets in the pair *oe43* are similar since they are sharing the three hashtags '#bpoil', '#bp', and '#oilspill'. This example shows that hashtags alone are not very indicative and useful for detecting redundancy in Twitter.

6 Conclusions

In this paper we introduced the notion of linguistic redundancy in micro-blogs and the task of tweet redundancy detection. We also presented an editorial study showing that redundancy is pervasive in Twitter, and that methods for its detection will be key in

[4] This is just a convention. Results would be the same by taking non-redundant pairs as the positive class.

[5] In o130, the common topic is 'farrah fawcett': "farrah fawcett not recognized at the Oscars memorial?" and "snubbed farrah fawcett. #oscars" are used by the annotators to make the decision.

gs	FOR+WBOW	WBOW	BOW	id	t_1	t_2
R	11	137	130	*o165*	"is that True that johnny depp died???"	"Rip johnny depp? This cannot be True"
R	32	246	239	*o942*	"sad...jim carrey and jenny mccarthy have called it quits..."	"jim carrey & jenny mccarthy broke up! omg! bummer! they were the cutest crazy couple ever."
R	43	165	158	*o130*	"farrah fawcett & bea arthur not recognized at the Oscars memorial? really?"	"i dont understand how they included michael jackson in the memorial tribute as an actor but snubbed farrah fawcett. #oscars"
R	101	632	641	*o21*	"Oscars forgot farrah fawcett??"	"farrah fawcett snubbed at Oscars appeared in a movie with best actor Jeff Bridges... disgusting"
N	467	161	155	*oe387*	"We may die in volcanic ash today. Choose your final pose soon to look cool for future archaeologists. #ashtag"	"# Just heard about the Icelandic volcanic ash thing, not really interested but it has the best hashtag ever, #ashtag !"
N	572	96	92	*oe43*	"Many Endangered Turtles Dying On Texas Gulf Coast http://ow.ly/1FbB8 via @nprnews #bpoil #bp #oilspill"	"Species Most at Risk Because of the Oil Spill http://ow.ly/1FcB7 #bpoil #bp #oilspill"
N	614	129	124	*oe64*	"http://bit.ly/d8W7Xw #ashtag IN PICTURES: Icelandic volcanic eruption"	"So, who's going to take a crack at pronouncing the part of Iceland the eruption was in? #ashtag"

Table 3: Ranks of some tweet pairs according to the scores of the different classifiers.

the future for the development of accurate Twitter-based applications. In the second part of the paper we presented some promising models for redundancy detection that show encouraging results when compared to typical lexical baselines. Even with the ungrammaticalities used in tweets, syntactic feature spaces are effective in modeling redundancy, especially when used in first-order rules.

In future work we plan to improve our system by adapting existing linguistic tools and resources to Twitter (e.g. syntactic parsers). We also plan to investigate the use of semantic roles and contextual information to improve the models. For example, the tweets that other users post about the same topic of the target-pair may be of some help. Finally, we are investigating the integration of our models into real applications such a the enrichment of news articles with related and *diverse* content from social media.

References

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Posters Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 36–44, Beijing, China.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, Washington.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 263–270.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18, Ann Arbor, Michigan.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In Quionero-Candela et al., editor, *LNAI 3944: MLCW 2005*, pages 177–190, Milan, Italy. Springer-Verlag.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Posters Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 241–249, Beijing, China.

Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D. Manning. 2006. Learning to distinguish valid textual entailments. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, Venice, Italy.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics (Coling 2004)*, pages 350–356, Geneva, Switzerland.

Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. 2010. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 295–303, Beijing, China.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado.

Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via graph matching. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 387–394, Vancouver, British Columbia, Canada.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with LCC's groundhog system. In Bernardo Magnini and Ido Dagan, editors, *Proceedings of the 2nd PASCAL RTE Challenge*, Venice, Italy.

Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. 2007. Why we Twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007*.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th International Conference on Research in Computational Linguistics ROCLING*, pages 132–139, Tapei, Taiwan.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods-Support Vector Learning*. MIT Press.

Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. 2008. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*, pages 19–24, Seattle, WA, USA.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is twitter, a social network or a news media? In *Proceedings of WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, Raleigh, North Carolina, USA.

Cindy-Xide Lin, Bo Zhao, Qiaozhu Mei, and Jiawei Han. 2010. Pet: a statistical model for popular events tracking in social communities. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 929–938, Washington, DC, USA.

Xiaohua Liu, Kuan Li, Bo Han, Ming Zhou, Long Jiang, Zhongyang Xiong, and Changning Huang. 2010. Semantic role labeling for news tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 698–706, Beijing, China, August.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 41–48, New York City, USA.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, November.

Alessandro Moschitti and Fabio Massimo Zanzotto. 2007. Fast and effective kernels for relational learning from texts. In *Proceedings of the International Conference of Machine Learning (ICML)*, Corvallis, Oregon.

Eamonn Newman, Nicola Stokes, John Dunnion, and Joe Carthy. 2005. Textual entailment recognition using a linguistically-motivated decision tree classifier. In Joaquin Quiñonero Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *MLCW*, volume 3944 of *Lecture Notes in Computer Science*, pages 372–384. Springer.

Sebastian Padó, 2006. *User's guide to* `sigf`*: Significance testing by approximate randomisation*.

Pear-Analytics. 2009. Twitter study - august 2009.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, pages 38–41, Boston, MA.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 181–189, Los Angeles, California.

Ana-Maria Popescu and Marco Pennacchiotti. 2010. Detecting controversial events from twitter. In *In Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1873–1876.

Daniel Ramage, Susan Dumais, and Dan Liebling. 2010. Characterizing microblogs with topic models. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 130–137.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, Los Angeles, California.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics*, pages 947–953, Morristown, NJ, USA.

Fabio Massimo Zanzotto and Lorenzo Dell'Arciprete. 2009. Efficient kernels for sentence pair classification. In *Conference on Empirical Methods on Natural Language Processing*, pages 91–100, 6-7 August.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the 21st Coling and 44th ACL*, pages 401–408, Sydney, Australia, July.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15-04:551–582.

Q. Zhao, P. Mitra, and B. Chen. 2007. Temporal and information flow based event detection from social text streams. In *Proceedings of the 22nd national conference on Artificial intelligence*, pages 1501–1506, Vancouver, British Columbia, Canada.

Divide and Conquer: Crowdsourcing the Creation of Cross-Lingual Textual Entailment Corpora

Matteo Negri
FBK-irst
Trento, Italy
negri@fbk.eu

Luisa Bentivogli
FBK-irst
Trento, Italy
bentivo@fbk.eu

Yashar Mehdad
FBK-irst and University of Trento
Trento, Italy
mehdad@fbk.eu

Danilo Giampiccolo
CELCT
Trento, Italy
giampiccolo@celct.it

Alessandro Marchetti
CELCT
Trento, Italy
amarchetti@celct.it

Abstract

We address the creation of cross-lingual textual entailment corpora by means of crowdsourcing. Our goal is to define a cheap and replicable data collection methodology that minimizes the manual work done by expert annotators, without resorting to preprocessing tools or already annotated monolingual datasets. In line with recent works emphasizing the need of large-scale annotation efforts for textual entailment, our work aims to: *i)* tackle the scarcity of data available to train and evaluate systems, and *ii)* promote the recourse to crowdsourcing as an effective way to reduce the costs of data collection without sacrificing quality. We show that a complex data creation task, for which even experts usually feature low agreement scores, can be effectively decomposed into simple subtasks assigned to non-expert annotators. The resulting dataset, obtained from a pipeline of different jobs routed to Amazon Mechanical Turk, contains more than 1,600 aligned pairs for each combination of texts-hypotheses in English, Italian and German.

1 Introduction

Cross-lingual Textual Entailment (CLTE) has been recently proposed by (Mehdad et al., 2010; Mehdad et al., 2011) as an extension of Textual Entailment (Dagan and Glickman, 2004). The task consists of deciding, given a text (T) and an hypothesis (H) *in different languages*, if the meaning of H can be inferred from the meaning of T. As in other NLP applications, both for monolingual and cross-lingual TE, the availability of large quantities of annotated data is an enabling factor for systems development and evaluation. Until now, however, the scarcity of such data on the one hand, and the costs of creating new datasets of reasonable size on the other, have represented a bottleneck for a steady advancement of the state of the art.

In the last few years, monolingual TE corpora for English and other European languages have been created and distributed in the framework of several evaluation campaigns, including the RTE Challenge[1], the Answer Validation Exercise at CLEF[2], and the Textual Entailment task at EVALITA[3]. Despite the differences in the design of the tasks, all the released datasets were collected through similar procedures, always involving expensive manual work done by expert annotators. Moreover, in the data creation process, large amounts of hand-crafted T-H pairs often have to be discarded in order to retain only those featuring full agreement, in terms of the assigned entailment judgements, among multiple annotators. The amount of discarded pairs is usually high, contributing to increase the costs of creating textual entailment datasets[4].

The issues related to the shortage of datasets and the high costs for their creation are more evident

[1] http://www.nist.gov/tac/2011/RTE/

[2] http://nlp.uned.es/clef-qa/ave/

[3] http://www.evalita.it/2009/tasks/te

[4] For instance, in the first five RTE Challenges, the average effort needed to create 1,000 pairs featuring full agreement among 3 annotators was around 2.5 person-months. Typically, around 25% of the original pairs had to be discarded during the process, due to low inter-annotator agreement (Bentivogli et al., 2009).

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 670–679,
Edinburgh, Scotland, UK, July 27–31, 2011.

in the CLTE scenario, where: *i)* the only dataset currently available is an English-Spanish corpus obtained by translating the RTE-3 corpus (Negri and Mehdad, 2010), and *ii)* the application of the standard methods adopted to build RTE pairs requires proficiency in multiple languages, thus significantly increasing the costs of the data creation process.

To address these issues, in this paper we devise a cost-effective methodology to create cross-lingual textual entailment corpora. In particular, we focus on the following problems:

(1) Is it possible to collect T-H pairs minimizing the intervention of expert annotators? To address this question, we explore the feasibility of crowdsourcing the corpus creation process. As a contribution beyond the few works on TE/CLTE data acquisition, we define an effective methodology that: *i)* does not involve experts in the most complex (and costly) stages of the process, *ii)* does not require preprocessing tools, and *iii)* does not rely on the availability of already annotated RTE corpora.

(2) How can we guarantee good quality of the collected data at a low cost? We address the quality control issue through the decomposition of a complex task (*i.e.* creating and annotating entailment pairs) into smaller sub-tasks. Complex tasks are usually hard to explain in a simple way understandable to non-experts, difficult to accomplish, and not suitable for the application of the quality-check mechanisms provided by current crowdsourcing services. Our "divide and conquer" solution represents the first attempt to address a complex task involving content *generation* and *labelling* through the definition of a cheap and reliable pipeline of simple tasks which are easy to define, accomplish, and control.

(3) Can we adapt such methodology to collect cross-lingual T-H pairs? We tackle this question by separating the problem of creating and annotating TE pairs from the issues related to the multilingual dimension. Our solution builds on the assumption that entailment annotations can be projected across aligned T-H pairs in different languages. In this case, a complex multilingual task is reduced to a sequence of simpler subtasks where the most difficult one, the generation of entailment pairs, is entirely monolingual. Besides ensuring cost-effectiveness, our solution allows us to overcome the problem of finding workers that are proficient in multiple languages. Moreover, since the core monolingual tasks of the process are carried out by manipulating English texts, we are able to address the very large community of English speaking workers, with a considerable reduction of costs and execution time. Finally, as a by-product of our method, the acquired pairs are fully aligned for all language combinations, thus enabling meaningful comparisons between scenarios of different complexity (monolingual TE, and CLTE between close or distant languages).

We believe that, in the same spirit of recent works promoting large-scale annotation efforts around entailment corpora (Sammons et al., 2010; Bentivogli et al., 2010), the proposed approach and the resulting dataset[5] will contribute to meeting the strong need for resources to develop and evaluate novel solutions for textual entailment.

2 Related Works

Crowdsourcing services, such as Amazon Mechanical Turk[6] (MTurk) and CrowdFlower[7], have been recently used with success for a variety of NLP applications (Callison-Burch and Dredze, 2010). The idea is that the acquisition and annotation of large amounts of data needed to train and evaluate NLP tools can be carried out in a cost-effective manner by defining simple Human Intelligence Tasks (HITs) routed to a crowd of non-expert workers (aka "Turkers") hired through on-line marketplaces.

As regards textual entailment, the first work exploring the use of crowdsourcing services for data *annotation* is described in (Snow et al., 2008), which shows high agreement between non-expert annotations of the RTE-1 dataset and existing gold standard labels assigned by expert labellers.

Focusing on the actual *generation* of monolingual entailment pairs, (Wang and Callison-Burch, 2010) experiments the use of MTurk to collect facts and counter facts related to texts extracted from an existing RTE corpus annotated with named entities. Taking a step beyond the task of annotating exist-

[5]The CLTE corpora described in this paper will be made freely available for research purposes through the website of the funding EU Project CoSyne (http://www.cosyne.eu/).

[6]https://www.mturk.com/

[7]Although MTurk is directly accessible only to US citizens, the CrowdFlower service (http://crowdflower.com/) provides an interface to MTurk for non-US citizens.

ing datasets, and showing the feasibility of involving non-experts also in the generation of TE pairs, this approach is more relevant to our objectives. However, at least two major differences with our work have to be remarked. First, they still use available RTE data to obtain a monolingual TE corpus, whereas we pursue the more ambitious goal of generating from scratch aligned CLTE corpora for different language combinations. To this aim, we do not resort to already annotated data, nor language-specific preprocessing tools. Second, their approach involves qualitative analysis of the collected data only *a posteriori*, after manual removal of invalid and trivial generated hypotheses. In contrast, our approach integrates quality control mechanisms at all stages of the data collection/annotation process, thus minimizing the recourse to experts to check the quality of the collected material.

Related research in the CLTE direction is reported in (Negri and Mehdad, 2010), which describes the creation of an English-Spanish corpus obtained from the RTE-3 dataset by translating the English hypotheses into Spanish. Translations have been crowdsourced adopting a methodology based on translation-validation cycles, defined as separate HITs. Although simplifying the CLTE corpus creation problem, which is recast as the task of translating already available annotated data, this solution is relevant to our work for the idea of combining gold standard units and "validation HITS" as a way to control the quality of the collected data at runtime.

3 Quality Control of Crowdsourced Data

The design of data acquisition HITs has to take into account several factors, each having a considerable impact on the difficulty of instructing the workers, the quality and quantity of the collected data, the time and overall costs of the acquisition. A major distinction has to be made between jobs requiring data *annotation*, and those involving content *generation*. In the former case, Turkers are presented with the task of labelling input data referring to a fixed set of possible values (*e.g.* making a choice between multiple alternatives, assigning numerical scores to rank the given data). In the latter case, Turkers are faced with creative tasks consisting in the production of textual material (*e.g.* writing a correct translation, or a summary of a given text).

The ease of controlling the quality of the acquired data depends on the nature of the job. For annotation jobs, quality control mechanisms can be easily set up by calculating Turkers' agreement, by applying voting schemes, or by adding hidden gold units to the data to be annotated[8]. In contrast, the quality of the results of content generation jobs is harder to assess, due to the fact that multiple valid results are acceptable (*e.g.* the same content can be expressed, translated, or summarized in different ways). In such situations the standard quality control mechanisms are not directly applicable, and the detection of errors requires either costly manual verification at the end of the acquisition process, or more complex and creative solutions integrating HITs for quality check.

Most of the approaches to content generation proposed so far rely on *post hoc* verification to filter out undesired low-quality data (Mrozinski et al., 2008; Mihalcea and Strapparava, 2009; Wang and Callison-Burch, 2010). The few solutions integrating validation HITs address the translation of single sentences, a task that is substantially different from ours (Negri and Mehdad, 2010; Bloodgood and Callison-Burch, 2010). Compared to sentence translation, the task of creating CLTE pairs is both harder to explain without recurring to notions that are difficult to understand to non-experts (*e.g.* "semantic equivalence", "unidirectional entailment"), and harder to execute without mastering these notions. To tackle these issues the "divide and conquer" approach described in the next section consists in the decomposition of a difficult *content generation* job into easier subtasks that are: *i)* self-contained and easy to explain, *ii)* easy to execute without any NLP expertise, and *iii)* suitable for the integration of a variety of runtime control mechanisms (regional qualifications, gold units, "validation HITs") able to ensure a good quality of the collected material.

[8]Both MTurk and CrowdFlower provide means to check workers' reliability, and weed out untrusted ones without money waste. These include different types of qualification mechanisms, the possibility of giving work only to known trusted Turkers (only with MTurk), and the possibility of adding hidden gold standard units in the data to be annotated (offered as a built-in mechanism only by CrowdFlower).

4 CLTE Corpus Creation Methodology

Our approach builds on a pipeline of HITs routed to MTurk's workforce through the CrowdFlower interface. The objective is to collect aligned T-H pairs for different language combinations, reproducing an RTE-like annotation style. However, our annotation is not limited to the standard RTE framework, where only unidirectional entailment from T to H is considered. As a useful extension, we annotate any possible entailment relation between the two text fragments, including: *i)* bidirectional entailment (*i.e.* semantic equivalence between T and H), *ii)* unidirectional entailment from T to H, and *iii)* unidirectional entailment from H to T. The resulting pairs can be easily used to generate not only standard RTE datasets[9], but also general-purpose collections featuring multi-directional entailment relations.

4.1 Data Acquisition and Annotation

We collect large amounts of CLTE pairs carrying out the most difficult part of the process (the creation of entailment-annotated pairs) at a monolingual level. Starting from a set of parallel sentences in *n* languages, (*e.g.* L1, L2, L3), *n* entailment corpora are created: *one* monolingual (L1/L1), and *n-1* cross-lingual (L1/L2, and L1/L3).

The monolingual corpus is obtained by modifying the sentences only in one language (L1). Original and modified sentences are then paired and annotated to form an entailment dataset for L1. The CLTE corpora are obtained by combining the modified sentences in L1 with the original sentences in L2 and L3, and projecting to the multilingual pairs the annotations assigned to the monolingual pairs.

In principle, only two stages of the process require crowdsourcing multilingual tasks, but do not concern entailment annotations. The first one, at the beginning of the process, aims to obtain a set of parallel sentences to start with, and can be done in different ways (*e.g.* crowdsourcing the translation of a set of sentences). The second one, at the end of the process, consists of translating the modified L1 sentences into other languages (*e.g.* L2) in order to extend the corpus to cover new language combinations (*e.g.* L2/L2, L2/L3).

The execution of the two "multilingual" stages is not strictly necessary but depends on: *i)* the availability of parallel sentences to start the process, and *ii)* the actual objectives in terms of language combinations to be covered[10].

As regards the first stage, in this work we started from a set of 467 English/Italian/German aligned sentences extracted from parallel documents downloaded from the Cafebabel European Magazine[11]. Concerning the second multilingual stage, we performed only one round of translations from English to Italian to extend the *3* combinations obtained without translations (ENG/ENG, ENG/ITA, and ENG/GER) with the new language combinations ITA/ITA, ITA/ENG, and ITA/GER.

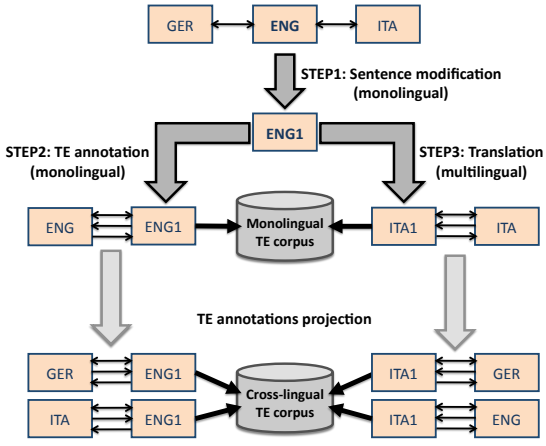


Figure 1: Corpus creation process.

The main steps of our corpus creation process, depicted in Figure 1, can be summarized as follows:

Step1: Sentence modification. The original English sentences (ENG) are modified through (monolingual) *generation* HITs asking Turkers to: *i)* preserve the meaning of the original sentences using different surface forms, or *ii)* slightly change their meaning by adding or removing content. Our assumption, in line with (Bos et al., 2009), is that

[9] With the positive examples drawn from bidirectional and unidirectional entailments from T to H, and the negative ones drawn from unidirectional entailments from H to T.

[10] Starting from parallel sentences in *n* languages, the *n* corpora obtained without recurring to translations can be augmented, by means of translation HITs, to create the full set of language combinations. Each round of translation adds 1 monolingual corpus, and *n-1* CLTE corpora.

[11] http://www.cafebabel.com/

another way to think about entailment is to consider whether one text *T1* adds new information to the content of another text *T*: if so, then *T* is entailed by *T1*.

The result of this phase is a set of texts (ENG1) that can be of three types:

1. Paraphrases of the original ENG texts, that will be used to create bidirectional entailment pairs (ENG↔ENG1);

2. More specific sentences (the outcome of content addition operations), used to create ENG←ENG1 unidirectional entailment pairs;

3. More general sentences (the outcome of content removal operations), used to create ENG→ENG1 unidirectional entailment pairs.

Step2: TE Annotation. Entailment pairs composed of the original sentences (ENG) and the modified ones (ENG1) are used as input of (monolingual) *annotation* HITs asking Turkers to decide which of the two texts contains more information. As a result, each ENG/ENG1 pair is annotated as an example of uni-/bidirectional entailment, and stored in the monolingual English corpus. Since the original ENG texts are aligned with the ITA and GER texts, the entailment annotations of ENG/ENG1 pairs can be projected to the other language pairs and the ITA/ENG1 and GER/ENG1 pairs are stored in the CLTE corpus. The possibility of projecting TE annotations is based on the assumption that the semantic information is mostly preserved during the translation process. This particularly holds at the denotative level (i.e. regarding the truth values of the sentence) which is crucial to semantic inference. At other levels (e.g. lexical) there might be slight semantic variations which, however, are very unlikely to play a crucial role in determining entailment relations.

Step3: Translation. The modified sentences (ENG1) are translated into Italian (ITA1) through (multilingual) *generation* HITs reproducing the approach described in (Negri and Mehdad, 2010). As a result, three new datasets are produced by automatically projecting annotations: the monolingual ITA/ITA1, and the cross-lingual ENG/ITA1 and GER/ITA1.

Since the solution adopted for sentence translation does not present novelty factors, the remainder of this paper will omit further details on it. Instead, the following sections will focus on the more challenging tasks of sentence modification and TE annotation.

4.2 Crowdsourcing Sentence Modification and TE Annotation

Sentence modification and TE annotation have been decomposed into a pipeline of simpler monolingual English sub-tasks. Such pipeline, depicted in Figure 2, involves several types of generation/annotation HITs designed to be easily understandable to non-experts. Each HIT consists of: *i)* a set of instructions for a specific task (*e.g.* paraphrasing a text), *ii)* the data to be manipulated (*e.g.* an English sentence), and *iii)* a test to check workers' reliability. To cope with the quality control issues discussed in Section 3, such tests are realized using gold standard units, either hidden in the data to be annotated (annotation HITs) or defined as test questions that workers must correctly answer (generation HITs). Moreover, regional qualifications are applied to all HITs. As a further quality check, all the annotation HITs consider Turkers' agreement as a way to filter out low quality results (only annotations featuring agreement among 4 out of 5 workers are retained). The six HITs defined for each subtask can be described as follows:

1. Paraphrase (generation). Modify an English text (ENG), in order to produce a semantically equivalent variant (ENG1). As a reliability test, before creating the paraphrase workers are asked to judge if two English sentences contain the same information.

2. Grammaticality (annotation). Decide if an English sentence is grammatically correct. This validation HIT represents a quality check of the output of each generation task (*i.e.* paraphrasing, and add/remove information HITs).

3. Bidirectional Entailment (annotation). Decide whether two English sentences, the original ENG and the modified ENG1, contain the same information (*i.e.* are semantically equivalent).

4a. Add Information (generation). Modify an English text to create a more specific one by adding content. As a reliability test, before generating the

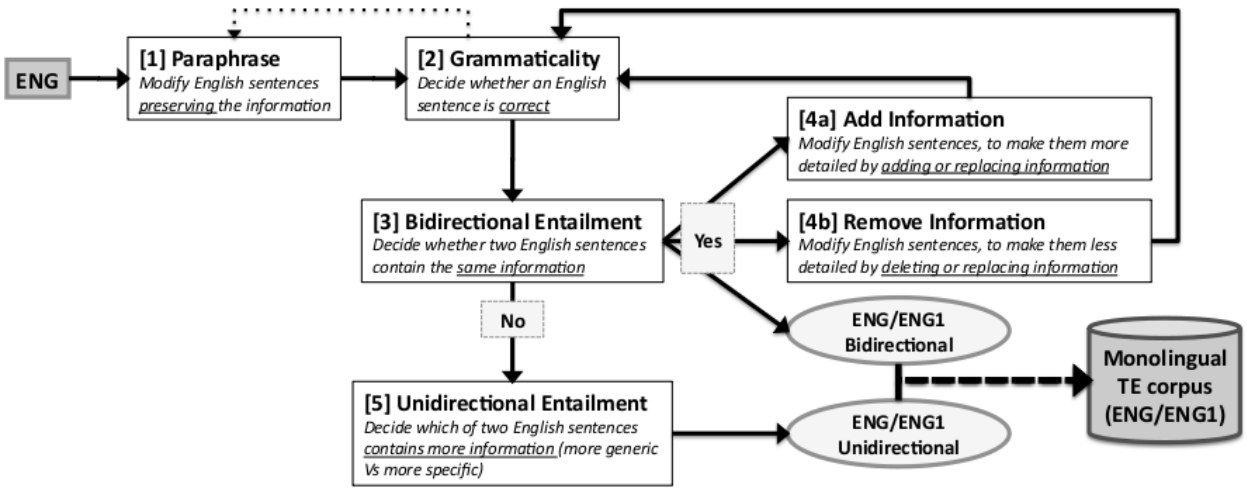


Figure 2: Sentence modification and TE annotation pipeline.

new sentence workers are asked to judge which of two given English sentences is more detailed.

4b. Remove Information (generation). Modify an English text to create a more general one by removing part of its content. As a reliability test, before generating the new sentence workers are asked to judge which of two given English sentences is less detailed.

5. Unidirectional Entailment (annotation). Decide which of two English sentences (the original ENG, and a modified ENG1) provides more information.

These HITs are combined in an iterative process that alternates text generation, grammaticality check, and entailment annotation steps. As a result, for each original ENG text we obtain multiple ENG1 variants of the three types (paraphrases, more general texts, and more specific texts) and, in turn, a set of annotated monolingual (ENG/ENG1) TE pairs.

As described in Section 4.1, the resulting monolingual English TE corpus (ENG/ENG1) is used to create the following mono/cross-lingual TE corpora:

- ITA/ENG1, and GER/ENG1 (by projecting TE annotations)
- ITA/ITA1, GER/ITA1, and ENG/ITA1 (by translating the ENG1 texts into Italian, and projecting TE annotations)

5 The Resulting CLTE Corpora

This section provides a quantitative and qualitative analysis of the results of our corpus creation methodology, focusing on the collected ENG-ENG1 monolingual dataset. It has to be remarked that, as an effect of the adopted methodology, all the observations and the conclusions drawn hold for the collected CLTE corpora as well.

5.1 Quantitative Analysis

Table 1 provides some details about each step of the pipeline shown in Figure 2. For each HIT the table presents: *i)* the number of items (sentences, or pairs of sentences) given in input, *ii)* the number of items (sentences or annotations) produced as output, *iii)* the number of items discarded when the agreement threshold was not reached, *iv)* the number of entailment pairs added to the corpus, *v)* the time (days and hours) required by the MTurk workforce to complete the job, and *vi)* the cost of the job.

In **HIT-1** (Paraphrase) 1,414 paraphrases were collected asking three different meaning-preserving modifications of each of the 467 original sentences[12]. From a practical point of view, such redundancy aims to ensure a sufficient number of grammatically correct and semantically equivalent modified sentences. From a theoretical point of view,

[12]Often, crowdsourced jobs return a number of output items that is slightly larger than required, due to the labour distribution mechanism internal to MTurk.

HIT	# Input items	# Output items	# Discarded items	# Pairs to corpus	MTurk time	Cost ($)
1. Paraphrase	467	1,414			5d+10.5h	45.48
2. Grammaticality	1,414	1,326	88 (6.22%)		1d+15h	56.88
3. Bidirectional Ent.	1,326	1,213 (yes=1,205 no=8)	113 (8.52%)	301	3d+2h	53.47
4a. Add Info	452	916			3d	37.02
4b. Remove Info	452	923			2d+22h	29.73
2. Grammaticality	1,839	1,749	90 (4.89%)		2d+5h	64.37
3. Bidirectional Ent.	1,749	1,438 (yes=148 no=1,290)	311 (17.78%)	148	3d+20.5h	70.52
5. Unidirectional Ent.	1,298	1,171	127 (9.78%)	1,171 (491 + 680)	8.5h	78.24
TOTAL			721	**1,620**	**22d+11h**	**435.71**

Table 1: The monolingual dataset creation pipeline.

collecting many variants of a small pool of original sentences aims to create pairs featuring different entailment relations with similar superficial forms. This, in principle, should allow to obtain a dataset which requires TE systems to focus more on deeper semantic phenomena than on the surface realization of the pairs.

The collected paraphrases were sent as input to **HIT-2** (Grammaticality). After this validation HIT, the number of acceptable paraphrases was reduced to 1,326 (with 88 discarded sentences, corresponding to 6.22% of the total).

The retained paraphrases were paired with their corresponding original sentences, and sent to **HIT-3** (Bidirectional Entailment) to be judged for semantic equivalence. The pairs marked as bidirectional entailments (1,205) were divided in three groups: 25% of the pairs (301) were directly stored in the final corpus, while the ENG1 paraphrases of the remaining 75% (904) were equally distributed to the next modification steps.

In both **HIT-4a** (Add Information) and **HIT-4b** (Remove information) two new modified sentences were asked for each of the 452 paraphrases received as input. The sentences collected in these generation tasks were respectively 916 and 923.

The new modified sentences were sent back to **HIT-2** (Grammaticality) and **HIT-3** (Bidirectional Entailment). As a result 1,438 new pairs were created; out of these, 148 resulted to be bidirectional entailments and were stored in the corpus.

Finally, the 1,298 entailment pairs judged as non-bidirectional in the two previously completed HIT-3 (8+1,290) were given as input to **HIT-5** (Unidirectional Entailment). The pairs which passed the agreement threshold were classified according to the judgement received, and stored in the corpus as unidirectional entailment pairs.

The analysis of Table 1 allows to formulate some considerations. First, the percentage of discarded items confirms the effectiveness of decomposing complex generation tasks into simpler subtasks that integrate validation HITs and quality checks based on non-experts' agreement. In fact, on average, around 9.5% of the generated items were discarded without experts' intervention[13]. Second, the amount of discarded items gives evidence about the relative difficulty of each HIT. As expected, we observe lower rejection rates, corresponding to higher inter-annotator agreement, for grammaticality HITs (5.55% on average) than for more complex entailment-related tasks (12.02% on average).

Looking at costs and execution time, it is hard to draw definite conclusions due to several factors that influence the progress of the crowdsourced jobs (*e.g.* the fluctuations of Turkers' performances, the time of the day at which jobs are posted, the difficulty to set the optimal cost for a given HIT[14]). On the one hand, as expected, the more creative "Add Info" task proved to be more demanding than the "Remove Info": even though it was paid more,

[13]Moreover, it is worthwhile noticing that around 20% of the collected items were automatically rejected (and not paid) due to failures on the gold standard controls created both for generation and annotation tasks.

[14]The payment for each HIT was set on the basis of a previous feasibility study aimed at determining the best trade-off between cost and execution time. However, replicating our approach would not necessarily result in the same costs.

it still took little more time to be completed. On the other hand, although the "Unidirectional Entailment" task was expected to be more difficult and thus rewarded more than the "Bidirectional Entailment" one, in the end it took notably less time to be completed. Nevertheless, the overall figures (435 USD, and about 22.5 days of MTurk work to complete the process)[15] clearly demonstrate the effectiveness of the approach. Even considering the time needed for an expert to manage the pipeline (*i.e.* one week to prepare gold units, and to handle the I/O of each HIT), these figures show that our methodology provides a cheaper and faster way to collect entailment data in comparison with the RTE average costs reported in Section 1.

As regards the amount of data collected, the resulting corpus contains 1,620 pairs with the following distribution of entailment relations: *i)* 449 bidirectional entailments, *ii)* 491 ENG→ENG1 unidirectional entailments, and *iii)* 680 ENG←ENG1 unidirectional entailments.

It must be noted that our methodology does not lead to the creation of pairs where some information is provided in one text and not in the other, and viceversa, as Example 1 shows:

Example 1.
ENG: New theories were emerging in the field of psychology.
ENG1: New theories were rising, which announced a kind of veiled racism.

These negative examples in both directions represent a natural extension of the dataset, relevant also for specific application-oriented scenarios, and their creation will be addressed in future work.

Besides the achievement of our primary objectives, the adopted approach led to some interesting by-products. First, the generated corpora are perfectly suitable to produce entailment datasets similar to those used in the traditional RTE evaluation framework. In particular, considering any possible entailment relation between two text fragments, our annotation subsumes the one proposed in RTE campaigns. This allows for the cost-effective generation of RTE-like annotations from the acquired corpora by combining ENG↔ENG1 and ENG→ENG1 pairs to form 940 positive examples (449+491), keeping the 680 ENG←ENG1 as negative examples. Moreover, by swapping ENG and ENG1 in the unidirectional entailment pairs, 491 additional negative examples and 680 positive examples can be easily obtained.

Finally, the output of HITs 1-2-3 in Table 1 represents *per se* a valuable collection of 1,205 paraphrases. This suggests the great potential of crowdsourcing for paraphrase acquisition.

5.2 Qualitative Analysis

Through manual verification of more than 50% of the corpus (900 pairs), a total number of 53 pairs (5.9%) were found incorrect. The different errors were classified as follows:

Type 1: Sentence modification errors. Generation HITs are a minor source of errors, being responsible for 10 problematic pairs. These errors are either introduced by generating a false statement (Example 2), or by forming a not fully understandable, awkward, or non-natural sentence (Example 3).

Example 2.
ENG: Kosovo was the subject of major riots in 1989.
ENG1: The Russian city of Kosovo was the subject of ...

Example 3.
ENG: Balat is the Kurdish-Armenian district of Instanbul.
ENG1: Balat is a place, which is the Kurdish-Armenian ...

Type 2: TE annotation errors. The notion of containing more/less information, used in the "Unidirectional Entailment" HIT, can mostly be applied straightforwardly to the entailment definition. However, the concept of "more/less detailed", which generally works for factual statements, in some cases is not applicable. In fact, the MTurk workers have regularly interpreted the instructions about the amount of information as concerning the quantity of concepts contained in a sentence. This is not always corresponding to the actual entailment relation between the sentences. As a consequence, 43 pairs featuring wrong entailment annotations were encountered. These errors can be classified as follows:

a) 13 pairs, where the added/removed information changes the meaning of the sentence. In these cases, the modified sentence was judged more/less specific

[15] Although by projecting annotations the ENG1/ITA and ENG1/GER CLTE corpora came for free, the ITA1/ITA, ITA1/ENG, and ITA1/GER combinations created by crowdsourcing translations added 45 USD and approximately 5 days to these figures.

than the original one, leading to unidirectional entailment annotation. On the contrary, in terms of the standard entailment definition, the correct annotation is "no entailment" (as in Example 4, which was annotated as ENG→ENG1):

Example 4.
ENG: If you decide to live in Bulgaria, you have to like difficulties because they are not difficulties, they are challenges.
ENG1: You have to like difficulties as they are not difficulties, they are challenges.

b) 10 pairs where the incorrect annotation is due to a coreference problem, as in:

Example 5.
ENG: John Smith is the new CEO of the company.
ENG1: He is the new CEO of the company.

These pairs were labelled as unidirectional entailments (in the example above ENG→ENG1), under the assumption that a proper name is more specific and informative than a pronoun. However, adhering to the TE definition, co-referring expressions are equivalent, and their realization does not play any role in the entailment decision. This implies that the correct entailment annotation is "bidirectional".

c) 9 pairs where the sentences are semantically equivalent, but contain a piece of information which is explicit in one sentence, and implicit in the other. In these cases, Turkers judged the sentence containing the explicit mention as more specific, and thus the pair was annotated as unidirectional entailment.

Example 6.
ENG: I hear the click of the trigger and the burst of bullets reach me immediately.
ENG1: I hear the trigger and the burst of bullets reach me instantly.

In Example 6, the expression *"the trigger"* in ENG1 implicitly means *"the click of the trigger"*, making the two sentences equivalent, and the entailment bidirectional (instead of ENG→ENG1).

d) 7 pairs where the information removed from or added to the sentence is not relevant to the entailment relation. In these cases, the modified sentence was judged less/more specific than the original one (and thus considered as unidirectional entailment), even though the correct judgement is "bidirectional", as in:

Example 7.
ENG: At the same time, AKP is struggling with its approach to the EU.
ENG1: AKP is struggling with its approach to the European Union.

e) 4 pairs where the added/removed information concerns universally quantified general statements, about which the interpretation of "more/less specific" given by Turkers resulted in the wrong annotation.

Example 8.
ENG: I think the success of multicultural couples depends on the size of the cultural gap between the two partners
ENG1: I believe the success of the couples depends on the size of the cultural gap between the 2 partners.

In Example 8, the additional information (*"multicultural"*) restricts the set to which it refers (*"couples"*) making ENG entailed by ENG1, and not vice versa as resulted from Turkers' annotation.

In light of this analysis, we conclude that the sentence modification methodology proved to be successful, as the low number of Type 1 errors shows. Considering that the most expensive phase in the creation of a TE dataset is the generation of the pairs, this is a significant achievement. Differently, the entailment assessment phase appears to be more problematic, accounting for the majority of errors. As shown by Type 2 errors, this is due to a partial misalignment between the instructions given in our HITs, and the formal definition of textual entailment. For this reason, further experimentation will explore different ways to instruct workers (*e.g.* asking to consider proper names and pronouns as equivalent) in order to reduce the amount of errors produced. As a final remark, considering that in the creation of a TE dataset the manual check of the annotated pairs represents a minor cost, even the involvement of experts to filter out wrong annotations would not decrease the cost-effectiveness of the proposed methodology.

6 Conclusions

There is an increasing need of annotated data to develop new solutions to the Textual Entailment problem, explore new entailment-related tasks, and set up experimental frameworks targeting real-world applications. Following the recent trends promoting annotation efforts that go beyond the established RTE Challenge framework (unidirectional entailment between monolingual T-H pairs), in this

paper we addressed the multilingual dimension of the problem. Our primary goal was the creation of large-scale collections of entailment pairs for different language combinations. Besides that, we considered cost effectiveness and replicability as additional requirements. To achieve our objectives, we developed a "divide and conquer" methodology based on crowdsourcing. Our approach presents several key innovations with respect to the related works on TE data acquisition. These include the decomposition of a complex content generation task in a pipeline of simpler subtasks accessible to a large crowd of non-experts, and the integration of quality control mechanisms at each stage of the process. The result of our work is the first large-scale dataset containing both monolingual and cross-lingual corpora for several combinations of texts-hypotheses in English, Italian, and German. Among the advantages of our method it is worth mentioning: *i)* the full alignment between the created corpora, *ii)* the possibility to easily extend the dataset to new languages, and *iii)* the feasibility of creating general-purpose corpora, featuring multi-directional entailment relations, that subsume the traditional RTE-like annotation.

Acknowledgments

This work has been partially supported by the EC-funded project CoSyne (FP7-ICT-4-24853). The authors would like to thank Emanuele Pianta for the helpful discussions, and Giovanni Moretti for the valuable support in the creation of the CLTE dataset.

References

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. *Proceedings of TAC 2009*.

Luisa Bentivogli, Elena Cabrio, Ido Dagan, Danilo Giampiccolo, Medea Lo Leggio, and Bernardo Magnini. 2010. Building Textual Entailment Specialized Data Sets: a Methodology for Isolating Linguistic Phenomena Relevant to Inference. *Proceedings of LREC 2010*.

Michael Bloodgood and Chris Callison-Burch. 2010. Using Mechanical Turk to Build Machine Translation Evaluation Sets. *Proceedings of the NAACL 2010 Workshop on Creating Speech and Language Data With Amazons Mechanical Turk*.

Johan Bos, Fabio Massimo Zanzotto, and Marco Pennacchiotti. 2009. Textual Entailment at EVALITA 2009. *Proceedings of EVALITA 2009*.

Chris Callison-Burch and Mark Dredze. 2010. Creating Speech and Language Data With Amazons Mechanical Turk. *Proceedings NAACL-2010 Workshop on Creating Speech and Language Data With Amazons Mechanical Turk*.

Ido Dagan and Oren Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. *Proceedings of the PASCAL Workshop of Learning Methods for Text Understanding and Mining*.

Yashar Mehdad, Matteo Negri, and Marcello Federico. 2010. Towards Cross-Lingual Textual Entailment. *Proceedings of NAACL-HLT 2010*.

Yashar Mehdad, Matteo Negri, and Marcello Federico. 2011. Using Bilingual Parallel Corpora for Cross-Lingual Textual Entailment. *Proceedings of ACL-HLT 2011*.

Rada Mihalcea and Carlo Strapparava. 2009. The Lie Detector: Explorations in the Automatic Recognition of Deceptive Language. *Proceedings of ACL 2009*.

Joanna Mrozinski, Edward Whittaker, and Sadaoki Furui. 2008. Collecting a Why-Question Corpus for Development and Evaluation of an Automatic QA-System. *Proceedings of ACL 2008*.

Matteo Negri and Yashar Mehdad. 2010. Creating a Bilingual Entailment Corpus through Translations with Mechanical Turk: $100 for a 10-day Rush. *Proceedings of the NAACL 2010 Workshop on Creating Speech and Language Data With Amazons Mechanical Turk*.

Mark Sammons, V. G. Vinod Vydiswaran, and Dan Roth. 2010. Ask Not What Textual Entailment Can Do for You... *Proceedings of ACL 2010*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky and Andrew Y. Ng. 2008. Cheap and Fast - But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. *Proceedings of EMNLP 2008*.

Rui Wang and Chris Callison-Burch. 2010. Cheap Facts and Counter-Facts. *Proceedings of the NAACL 2010 Workshop on Creating Speech and Language Data With Amazons Mechanical Turk*.

Literal and Metaphorical Sense Identification through Concrete and Abstract Context

Peter D. Turney
Inst. for Info. Tech.
NRC Canada
Ottawa, Canada
peter.turney@nrc-cnrc.gc.ca

Yair Neuman
Dept. of Education
Ben-Gurion Univ.
Beer-Sheva, Israel
yneuman@bgu.ac.il

Dan Assaf
Dept. of Education
Ben-Gurion Univ.
Beer-Sheva, Israel
dan.assaf4@googlemail.com

Yohai Cohen
Gilasio Coding
Tel-Aviv, Israel
yohai@gilasio.com

Abstract

Metaphor is ubiquitous in text, even in highly technical text. Correct inference about textual entailment requires computers to distinguish the literal and metaphorical senses of a word. Past work has treated this problem as a classical word sense disambiguation task. In this paper, we take a new approach, based on research in cognitive linguistics that views metaphor as a method for transferring knowledge from a familiar, well-understood, or concrete domain to an unfamiliar, less understood, or more abstract domain. This view leads to the hypothesis that metaphorical word usage is correlated with the degree of abstractness of the word's context. We introduce an algorithm that uses this hypothesis to classify a word sense in a given context as either literal (denotative) or metaphorical (connotative). We evaluate this algorithm with a set of adjective-noun phrases (e.g., in *dark comedy*, the adjective *dark* is used metaphorically; in *dark hair*, it is used literally) and with the TroFi (Trope Finder) Example Base of literal and nonliteral usage for fifty verbs. We achieve state-of-the-art performance on both datasets.

1 Introduction

Metaphor is a natural consequence of our ability to reason by analogy (Gentner et al., 2001). It is so common in our daily language that we rarely notice it (Lakoff and Johnson, 1980). Identifying metaphorical word usage is important for reasoning about the implications of text.

Past work on the problem of distinguishing literal and metaphorical senses has approached it as a classical word sense disambiguation (WSD) task (Birke and Sarkar, 2006). Here, we take a different approach to the problem. Lakoff and Johnson (1980) argue that metaphor is a method for transferring knowledge from a concrete domain to an abstract domain. Therefore we hypothesize that the degree of abstractness in a word's context is correlated with the likelihood that the word is used metaphorically. This hypothesis is the basis for our algorithm for distinguishing literal and metaphorical senses.

Consider the following sentences:

L: He *shot down* my plane.
$\rightarrow C_1$: He *fired at* my plane.
$\nrightarrow A_1$: He *refuted* my plane.

M: He *shot down* my argument.
$\nrightarrow C_2$: He *fired at* my argument.
$\rightarrow A_2$: He *refuted* my argument.

The literal sense of *shot down* in L invokes knowledge from the domain of war. The metaphorical usage of *shot down* in M transfers knowledge from the concrete domain of war to the abstract domain of debate (Lakoff and Johnson, 1980).

The entailments of L and M depend on the intended senses of *shot down*. L entails the concrete *fired at* in C_1 (because, in order to literally shoot something down, you must first fire at it) but not the abstract *refuted* in A_1 (except perhaps as a joke). On the other hand, M entails *refuted* in A_2 but not *fired at* in C_2 (except perhaps as a novel metaphor).

In semiotics, Danesi (2003) argues that metaphor transfers *associations* from the source domain to the target domain. The metaphorical usage of *shot down* in M carries associations of violence and destruc-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 680–690,
Edinburgh, Scotland, UK, July 27–31, 2011.

tion that are not conveyed by A_2.

To make correct inferences about textual entailment, computers must be able to distinguish the literal and metaphorical senses of a word. Since recognizing textual entailment (RTE) is a core problem for NLP, with applications in Question Answering, Information Retrieval, Information Extraction, and Text Summarization, it follows that distinguishing literal and metaphorical senses is a problem for a wide variety of NLP tasks. The ability to recognize metaphorical word usage is a core requirement in the Intelligence Advanced Research Projects Activity (IARPA) Metaphor Program (Madrigal, 2011).[1]

Our approach to the problem of distinguishing literal and metaphorical senses is based on an algorithm for calculating the degree of abstractness of words. For instance, *plane* in L is rated 0.36396 (relatively concrete), whereas *argument* in M is rated 0.64617 (relatively abstract), which suggests that the verb *shot down* is used literally in L, whereas it is used metaphorically in M. Our abstractness rating algorithm is similar to Turney and Littman's (2003) algorithm for rating words according to their semantic orientation.

To classify a word usage as literal or metaphorical, based on the context, we use supervised learning with logistic regression. The abstractness rating algorithm is used to generate feature vectors from a word's context and training data is used to learn a logistic regression model that relates degrees of abstractness to the classes *literal* and *metaphorical*.

We evaluate our algorithm with three experiments. The first experiment involves one hundred adjective-noun phrases labeled *denotative* (literal) or *connotative* (metaphorical or nonliteral) by five annotators, according to the sense of the adjective.[2] For instance, *deep snow* is labeled *denotative* and *deep appreciation* is labeled *connotative*. The algorithm is able to predict the labels of the annotators with an average accuracy of 79%.

The next two experiments use the TroFi (Trope Finder) Example Base of literal and nonliteral usage for fifty verbs.[3] The fifty verbs occur in 3,737 sentences from The 1987-89 Wall Street Journal (WSJ) Corpus Release 1. In each sentence, the target verb is labeled *L* (literal) or *N* (nonliteral), according to the sense of the verb that is invoked by the sentence. A subset of twenty-five of the fifty verbs was used by Birke and Sarkar (2006).

In our second experiment, we duplicate the setup of Birke and Sarkar (2006) so that we can compare our results with theirs. In particular, a separate model is learned for each individual verb. We achieve an average f-score of 63.9%, compared to Birke and Sarkar's (2006) 64.9%.

In the third experiment, we train the algorithm on the twenty-five new verbs that were not used by Birke and Sarkar (2006) and then we test it on the old verbs. That is, the algorithm is tested with verbs that it has never seen before. The training verbs are merged to build a single model, instead of building a separate model for each individual verb. In this experiment, the average f-score is 68.1%.

The next section presents our algorithm for calculating the degree of abstractness of words. In Section 3, we review related work. The experiments are described in Section 4. We discuss the results of the experiments in Section 5 and conclude in Section 6.

2 Abstractness and Concreteness

Concrete words refer to things, events, and properties that we can perceive directly with our senses, such as *trees*, *walking*, and *red*.[4] Abstract words refer to ideas and concepts that are distant from immediate perception, such as *economics*, *calculating*, and *disputable*. In this section, we describe an algorithm that can automatically calculate a numerical rating of the degree of abstractness of a word on a scale from 0 (highly concrete) to 1 (highly abstract). For example, the algorithm rates *purvey* as 1, *donut* as 0, and *immodestly* as 0.5.

The algorithm is a variation of Turney and Littman's (2003) algorithm that rates words according to their semantic orientation. Positive semantic orientation indicates praise (*honest*, *intrepid*) and negative semantic orientation indicates criticism (*disturbing*, *superfluous*). The algorithm calculates the semantic orientation of a given word by comparing it to seven positive words and seven nega-

[1] See http://www.iarpa.gov/solicitations_metaphor.html.

[2] The labeled phrases are available from Yair Neuman.

[3] Available at http://www.cs.sfu.ca/ anoop/students/jbirke/.

[4] The word *red* has an abstract political sense, but our abstractness rating algorithm does not distinguish word senses. The more frequent concrete sense of *red* dominates, resulting in an abstractness rating of 0.24984 (highly concrete).

tive words that are used as paradigms of positive and negative semantic orientation:

Positive paradigm words: *good, nice, excellent, positive, fortunate, correct,* and *superior*.
Negative paradigm words: *bad, nasty, poor, negative, unfortunate, wrong,* and *inferior*.

Likewise, here we calculate the abstractness of a given word by comparing it to twenty abstract words and twenty concrete words that are used as paradigms of abstractness and concreteness.

Turney and Littman (2003) experimented with two measures of semantic similarity, pointwise mutual information (PMI) (Church and Hanks, 1989) and latent semantic analysis (LSA) (Landauer and Dumais, 1997). These measures take a pair of words as input and generate a numerical similarity rating as output. The semantic orientation of a given word is calculated as the sum of its similarity with the positive paradigm words minus the sum of its similarity with the negative paradigm words. Likewise, here we calculate the abstractness of a given word by the sum of its similarity with twenty abstract paradigm words minus the sum of its similarity with twenty concrete paradigm words. We then use a linear normalization to map the calculated abstractness value to range from 0 to 1.

Our algorithm for calculating abstractness uses a form of LSA to measure semantic similarity. This is described in detail in Section 2.1. Although Turney and Littman (2003) manually selected their fourteen paradigm words, here we use a supervised learning algorithm to choose our forty paradigm words, as explained in Section 2.2.

The MRC Psycholinguistic Database Machine Usable Dictionary (Coltheart, 1981) includes 4,295 words rated with degrees of abstractness by human subjects in psycholinguistic experiments.[5] The ratings range from 158 (highly abstract) to 670 (highly concrete). Table 1 gives some examples.

We used half of the 4,295 MRC words to train our supervised learning algorithm and the other half to validate the algorithm. On the testing set, the algorithm attains a correlation of 0.81 with the dictionary ratings. This indicates that the algorithm agrees well with human judgements of the degrees of abstractness of words.

[5] Available at http://ota.oucs.ox.ac.uk/headers/1054.xml.

Abstract Words	Rating	Concrete Words	Rating
as	158	ape	654
of	180	grasshopper	660
apt	183	tomato	662
however	186	milk	670

Table 1: Examples of abstract and concrete words from the MRC Dictionary (Coltheart, 1981).

2.1 Measuring Semantic Similarity

The variation of LSA that we use here is similar to Rapp's (2003) work. We modeled our similarity measure on Rapp's due to the high score of 92.5% that he achieved on a set of 80 multiple-choice synonym questions from the Test of English as a Foreign Language (TOEFL). The core idea is to represent words with vectors and calculate the similarity of two words by the cosine of the angle between the two corresponding vectors. The values of the elements in the vectors are derived from the frequencies of the words in a large corpus of text. This general approach is known as a Vector Space Model (VSM) of semantics (Salton et al., 1975).

We began with a corpus of 5×10^{10} words (280 gigabytes of plain text) gathered from university websites by a webcrawler.[6] We then indexed this corpus with the Wumpus search engine (Büttcher and Clarke, 2005).[7] We selected our vocabulary from the terms (words and phrases) in the WordNet lexicon.[8] By querying Wumpus, we obtained the frequency of each WordNet term in our corpus. We selected all WordNet terms with a frequency of 100 or more in our corpus. This resulted in a set of 114,501 terms. Next we used Wumpus to search for up to 10,000 phrases per term, where a phrase consists of the given term plus four words to the left of the term and four words to the right of the term. These phrases were used to build a word–context frequency matrix $\mathbf{F}$ with 114,501 rows and 139,246 columns. A row vector in $\mathbf{F}$ corresponds to a term in WordNet and the columns in $\mathbf{F}$ correspond to contexts (the words to the left and right of a given term in a given phrase) in which the term appeared.

The columns in $\mathbf{F}$ are unigrams (single words) in WordNet with a frequency of 100 or more in the corpus. A given unigram is represented by two

[6] Collected by Charles Clarke at the University of Waterloo.
[7] Wumpus is available at http://www.wumpus-search.org/.
[8] WordNet is available at http://wordnet.princeton.edu/.

columns, one marked *left* and one marked *right*. Suppose r is the term corresponding to the i-th row in $\mathbf{F}$ and c is the term corresponding to the j-th column in $\mathbf{F}$. Let c be marked *left*. Let f_{ij} be the cell in the i-th row and j-th column of $\mathbf{F}$. The numerical value in the cell f_{ij} is the number of phrases found by Wumpus in which the center term was r and c was the unigram closest to r on the left side of r. That is, f_{ij} is the frequency with which r was found in the context c in our corpus.

A new matrix $\mathbf{X}$, with the same number of rows and columns as in $\mathbf{F}$, was formed by calculating the Positive Pointwise Mutual Information (PPMI) of each cell in $\mathbf{F}$ (Turney and Pantel, 2010). The function of PPMI is to emphasize cells in which the frequency f_{ij} is statistically surprising, and hence particularly informative. This matrix was then smoothed with a truncated Singular Value Decomposition (SVD), which decomposes $\mathbf{X}$ into the product of three matrices $\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$. Finally, the terms were represented by the matrix $\mathbf{U}_k\mathbf{\Sigma}_k^p$, which has 114,501 rows (one for each term) and k columns (one for each latent contextual factor). The semantic similarity of two terms is given by the cosine of the two corresponding rows in $\mathbf{U}_k\mathbf{\Sigma}_k^p$. For more detail, see Turney and Pantel (2010).

There are two parameters in $\mathbf{U}_k\mathbf{\Sigma}_k^p$ that need to be set. The parameter k controls the number of latent factors and the parameter p adjusts the weights of the factors, by raising the corresponding singular values in $\mathbf{\Sigma}_k^p$ to the power p. The parameter k is well-known in the literature on LSA, but p is less familiar. The use of p was suggested by Caron (2001). Based on our past experience, we set k to 1000 and p to 0.5. We did not explore any alternative settings of these parameters for measuring abstractness.

2.2 Measuring Abstractness

Now that we have $\mathbf{U}_k\mathbf{\Sigma}_k^p$, all we need in order to measure abstractness is some paradigm words. We used the MRC Psycholinguistic Database Machine Usable Dictionary (Coltheart, 1981) to guide our search for paradigm words. We split the 4,295 MRC words into 2,148 for training (searching for paradigm words) and 2,147 for testing (evaluation of the final set of paradigm words). We began with an empty set of paradigm words and added words from the 114,501 rows of $\mathbf{U}_k\mathbf{\Sigma}_k^p$, one word at a time, alternating between adding a word to the concrete paradigm words and then adding a word to the abstract paradigm words. At each step, we added the paradigm word that resulted in the highest Pearson correlation with the ratings of the training words. This is a form of greedy forward search without backtracking. We stopped the search after forty paradigm words were found, in order to prevent overfitting of the training data.

Table 2 shows the forty paradigm words and the order in which they were selected. At each step, the correlation increases on the training set, but eventually it must decrease on the testing set. After forty steps, the training set Pearson correlation was 0.8600. At this point, we stopped the search for paradigm words and calculated the testing set Pearson correlation, which was 0.8064. This shows a small amount of overfitting of the training data. The testing set Spearman correlation was 0.8216.

For another perspective on the performance of the algorithm, we measured its accuracy on the testing set, by creating a binary classification task from the testing data. We calculated the median of the ratings of the 2,147 words in the test set. Every word with an abstractness above the median was assigned to the class 1 and every word with an abstractness below the median was assigned to the class 0. We then used the algorithm to guess the rating of each word in the test set, calculated the median guess, and likewise assigned the guesses to classes 1 and 0. The guesses were 84.65% accurate.

After generating the paradigm words with the training set and evaluating them with the testing set, we then used them to assign abstractness ratings to every term in the matrix. The result of this is that we now have a set of 114,501 terms (words and phrases) with abstractness ratings ranging from 0 to 1.[9] Based on the testing set performance, we estimate these 114,501 ratings would have a Pearson correlation of 0.81 with human ratings and an accuracy of 85% on binary (*abstract* or *concrete*) classification.

We chose to limit the search to forty paradigm words based on our past experience with semantic orientation (Turney and Littman, 2003). To validate this choice, we allowed the algorithm to continue

[9] The 114,501 rated terms are available from Peter Turney.

Concrete Paradigm Words			Abstract Paradigm Words		
Order	Word	Correlation	Order	Word	Correlation
1	donut	0.4447	2	sense	0.6165
3	antlers	0.6582	4	indulgent	0.6973
5	aquarium	0.7150	6	bedevil	0.7383
7	nursemaid	0.7476	8	improbable	0.7590
9	pyrethrum	0.7658	10	purvey	0.7762
11	swallowwort	0.7815	12	pigheadedness	0.7884
13	strongbox	0.7920	14	ranging	0.7973
15	sixth-former	0.8009	16	quietus	0.8067
17	restharrow	0.8089	18	regularisation	0.8123
19	recorder	0.8148	20	creditably	0.8188
21	sawmill	0.8212	22	arcella	0.8248
23	vulval	0.8270	24	nonproductive	0.8299
25	tenrecidae	0.8316	26	couth	0.8340
27	hairpiece	0.8363	28	repulsion	0.8400
29	sturnus	0.8414	30	palsgrave	0.8438
31	gadiformes	0.8451	32	goof-proof	0.8469
33	cobbler	0.8481	34	meshuga	0.8503
35	bullet	0.8521	36	dillydally	0.8538
37	dioxin	0.8550	38	reliance	0.8570
39	usa	0.8585	40	lumbus	0.8600

Table 2: The forty paradigm words and the Pearson correlation on the training set.

searching until one hundred paradigm words were found. This resulted in a training set Pearson correlation of 0.8963, but the testing set correlation was only 0.8097, which shows a significant amount of overfitting of the training data. Although the testing set correlation is slightly higher with one hundred paradigm words, we chose to base the following experiments on the forty paradigm words, because the difference between 0.8064 and 0.8097 is not significant, and the gap between the training and testing correlation (0.8963 versus 0.8097) indicates a problematic amount of overfitting. Furthermore, the execution time of the algorithm increases as the paradigm set increases.

We generated abstractness ratings for a large vocabulary of 114,501 words in order to maximize the variety of text genres and the range of applications for which our list of abstractness ratings would be useful. As a consequence of this large vocabulary, many of the words in Table 2 are rare and obscure; however, the measure of quality of the algorithm is the correlation with the testing set (0.81), not the familiarity of the words in the table. We include the table here so that other researchers can experiment with these paragidm words. The table may give some insight into the internal functioning of the algorithm, but the main output of the algorithm is the list of 114,501 words with abstractness ratings, not the list of paradigm words in Table 2.

3 Related Work

Here we discuss related work on metaphor and then work on measuring abstractness. As far as we know, our approach is the first in computational linguistics to bring these two themes together, although the connection is well-known in cognitive linguistics (Lakoff and Johnson, 1980) and cognitive psychology (Gentner et al., 2001).

3.1 Metaphor

The most closely related work is Birke and Sarkar's (2006) research on distinguishing literal and nonliteral usage of verbs. A later paper (Birke and Sarkar, 2007) provides more detail on their active learning system, briefly mentioned in the earlier paper. Birke and Sarkar (2006; 2007) treat the problem as a classical word sense disambiguation task (Navigli, 2009). A model is learned for each verb indepen-

dently from the other verbs. This approach cannot handle a new verb without additional training.

Hashimoto and Kawahara (2009) discuss work on a similar problem, distinguishing idiomatic usage from literal usage. They also approach this as a classical word sense disambiguation task. Idioms are somewhat different from metaphors, in that the meaning of an idiom (e.g., *kick the bucket*) is often difficult to derive from the meanings of the component words, unlike most metaphors.

Nissim and Markert (2003) use supervised learning to distinguish metonymic usage from literal usage. They take a classical WSD approach, learning a separate model for each target word. As with Birke and Sarkar (2006; 2007) and Hashimoto and Kawahara (2009), the core idea is to learn to classify word usage from similarity of context. Unlike these approaches, our algorithm generalizes beyond the specific semantic content of the context, paying attention only to the degrees of abstractness of the context.

Martin (1992) presents a knowledge-based approach to interpreting metaphors. This approach requires complex hand-coded rules, which are specific to a given domain (e.g., interpreting metaphorical questions from computer users, such as, "How can I *kill* a process?", in an online help system). The knowledge base cannot handle words that are not hand-coded in its rules and a new set of rules must be constructed for each new application domain.

Dolan (1995) describes an algorithm for extracting metaphors from a dictionary. Some suggestive examples are given, but the algorithm is not evaluated in any systematic way.

Mason (2004) takes a corpus-based approach to metaphor. His algorithm is based on a statistical approach to discovering the selectional restrictions of verbs. It then uses these restrictions to discover metaphorical mappings, such as, "Money flows like a liquid." Although the system can discover some metaphorical mappings, it was not designed to distinguish literal and metaphorical usages of words.

3.2 Abstractness

Changizi (2008) uses the hypernym hierarchy in WordNet to calculate the abstractness of a word. A word near the top of the hierarchy is considered abstract and a word near the bottom is considered concrete. It seems to us that the WordNet hypernym hierarchy captures the general–specific continuum, which might not be the same as the abstract–concrete continuum. It would be interesting to see how much correspondence there is between Changizi's measure of abstractness and the ratings in the MRC Psycholinguistic Database Machine Usable Dictionary (Coltheart, 1981). Also, note that adjectives and adverbs are outside of WordNet's hypernym hierarchy, and thus cannot be rated by Changizi's algorithm.

Xing et al. (2010) also use WordNet, but in a different way. They define the *concreteness* of a word sense (a WordNet synset) to be 1 if the given word sense is a hyponym of *physical entity* in the WordNet hypernym hierarchy; otherwise the *concreteness* is 0. We believe that, although physical entities are concrete, so are *redness* and *walking*, which are not hyponyms of *physical entity*. The category *physical entity* only partially captures concreteness.

4 Experiments

In the following experiments, we use the abstractness ratings of Section 2.2 to generate features for supervised machine learning. The learning algorithm we apply is logistic regression (Le Cessie and Van Houwelingen, 1992), as implemented in Weka (Witten and Frank, 2005).[10] In all experiments, we used the Weka parameter settings $R = 0.2$ (for robust ridge regression) and $M = -1$ (for unlimited iterations).

4.1 Adjectives

For this experiment, we selected five adjectives, *dark*, *deep*, *hard*, *sweet*, and *warm*. For each of the five adjectives, we identified twenty word pairs in which the first word is the adjective and the second word is a noun. These pairs were identified through the Corpus of Contemporary American English (COCA)[11] (Davies, 2009) by seeking the nouns that follow each adjective in the corpus and sorting the candidate adjective-noun pairs by frequency. We required a minimum pointwise mutual information (PMI) of 3 between the adjective and the noun. In some of the pairs, the adjective was

[10] Weka is available at http://www.cs.waikato.ac.nz/ml/weka/.

[11] Available at http://www.americancorpus.org/.

used in a denotative (literal) sense (*dark hair*) and in others it was used in a connotative (nonliteral) sense (*dark humor*). Table 3 gives some examples.

Adjective-Noun Pairs	Noun Abstractness
dark glasses	0.26826
dark chocolate	0.28211
dark energy	0.66207
dark mood	0.61858

Table 3: Some examples of adjective-noun pairs and the abstractness rating of the noun.

In this experiment, we used the abstractness rating of the noun (the context) to predict whether the adjective (the target) was used in a metaphorical or literal sense. Table 3 supports this idea, but it is easy to find counterexamples. Although *dark mood* is metaphorical, *bad mood* is literal. The difference is that *dark* has an abstractness rating of 0.43356 (relatively concrete), whereas *bad* has an abstractness rating of 0.63326 (relatively abstract). Metaphor results when a concrete word is imported into an abstract context (Lakoff and Johnson, 1980). Ideally, we should be comparing the abstractness of the target to the abstractness of the context. However, in our data, the target words are mostly concrete; thus we can focus on the context and ignore the target. We discuss this point further in Section 5.

Five judges, undergraduate students in psychology, were asked to judge whether the use of the adjective is a denotation or a connotation. The instructions were as follows:

> *Denotation* is the most direct or specific meaning of a word or expression while *connotation* is the meaning suggested by the word that goes beyond its literal meaning. For instance, the meaning of *bitter* is denotative in *bitter lemon* and connotative in *bitter relations*. In each of the following pairs, you will be asked to judge whether (1) the meaning of the first word is *denotative* or *connotative* and (2) to what extent it is denotative or connotative on a scale ranging from 1 to 4.

The judges were blind to the research hypothesis. Each judge received a booklet with the items organized by the groups of adjectives and presented in a random order. Overall, each subject was asked to evaluate one hundred pairs. Interjudge reliability was high, with Cronbach's Alpha equal to 0.95.

Our feature vectors for each pair contained only one element, the abstractness rating of the noun in the pair. We used logistic regression with ten-fold cross-validation to predict each judge's *denotative* and *connotative* labels. The results are summarized in Table 4. On average, we were able to predict a judge's labels with 79% accuracy.

Judge	Accuracy	Majority
1	0.730	0.590
2	0.810	0.570
3	0.840	0.560
4	0.790	0.510
5	0.780	0.520
Average	0.790	0.550

Table 4: The accuracy of logistic regression at predicting the labels of each judge.

Table 4 also shows the size of the majority class (the most common label) for each judge. For all of the judges, the accuracy was significantly greater than the size of the majority class (Fisher Exact test, 95% confidence level). The results support our hypothesis that the abstractness of the context is predictive of whether an adjective is used in a literal or metaphorical sense.

4.2 Known Verbs

For this experiment, we used the TroFi (Trope Finder) Example Base of literal and nonliteral usage for fifty verbs.[12] To compare our results with Birke and Sarkar's (2006) results, we use the same subset of twenty-five of the fifty verbs. These twenty-five verbs appear in 1,965 sentences, manually labeled *L* (literal) or *N* (nonliteral), according to the sense of the target verb. The verbs also appeared in some sentences labeled *U* (unannotated), but we ignored these sentences (although they could be useful for semi-supervised learning).

The label *nonliteral* is intended to be a broad category that includes *metaphorical* as a special case. Other types of nonliteral usage include *idiomatic* and *metonymical*, but it seems that most of the *nonliteral* cases in TroFi are in fact *metaphorical*, and

[12] Available at http://www.cs.sfu.ca/ anoop/students/jbirke/.

hence our hypothesis about the correlation of abstract context with metaphorical sense is appropriate for classifying the TroFi sentences.

Two examples of sentences from TroFi follow. Both contain the target verb *absorb*. The first sentence is *literal* and the second is *nonliteral*.

***L*:** An Energy Department spokesman says the sulfur dioxide might be simultaneously recoverable through the use of powdered limestone, which tends to *absorb* the sulfur.

***N*:** He said that MMWEC will have to *absorb* only $4 million in additional annual costs now paid by the Vermont utilities.

To generate feature vectors for the sentences, we first applied the OpenNLP part-of-speech tagger to the sentences.[13] We then looked for each word in our list of 114,501 abstractness ratings (Section 2.2). If the word was not found in the list, we applied the Morpha morphological analyzer to identify the stem of the word (e.g., the stem of *managing* is *manage*) (Minnen et al., 2001).[14] We then looked for the stem in our list. If it was still not found, we skipped it.

For each sentence, we created a vector with five features:

1. the average abstractness ratings of all nouns, excluding proper nouns
2. the average abstractness ratings of all proper nouns
3. the average abstractness ratings of all verbs, excluding the target verb
4. the average abstractness ratings of all adjectives
5. the average abstractness ratings of all adverbs

When there were no words for a given part of speech, we set the average to a default value of 0.5. Two examples of feature vectors follow, corresponding to the two TroFi sentences above.

***L*:** $\langle 0.3873, 0.5397, 0.6375, 0.2641, 0.5835 \rangle$

***N*:** $\langle 0.6120, 0.3726, 0.6699, 0.5612, 0.5000 \rangle$

The intuition here is that the weight of each context word, in predicting the class of the target verb, may depend on the part of speech of the context word. We leave it to the logistic regression algorithm to determine the appropriate weighting, based on the training data. (See Table 7 in the next section.)

Following Birke and Sarkar's (2006) approach, we treated each group of sentences for a given target verb as a separate learning problem. For each verb, we used ten-fold cross-validation to learn and test logistic regression models. To measure the performance of the models, we used three different scores, macro-averaged accuracy and two forms of macro-averaged f-score.

Birke and Sarkar (2006) explain their scoring as follows:

> *Literal recall* is defined as *(correct literals in literal cluster / total correct literals)*. *Literal precision* is defined as *(correct literals in literal cluster / size of literal cluster)*. If there are no literals, *literal recall* is 100%; *literal precision* is 100% if there are no nonliterals in the literal cluster and 0% otherwise. The *f-score* is defined as *(2 · precision · recall) / (precision + recall)*. Nonliteral precision and recall are defined similarly. Average precision is the average of literal and nonliteral precision; similarly for average recall. For overall performance, we take the f-score of average precision and average recall.

The overall score is a macro-average, in which each verb has equal weight, regardless of how many sentences it appears in.

Every verb in TroFi has at least one *literal* usage and one *nonliteral* usage, so there is no issue with the definition of recall as 100% when there are no literals or no nonliterals. However, we believe that the definition of precision as 100% when no sentence is assigned to the literal or nonliteral cluster gives too high a score to the trivial algorithm of always guessing the majority class. The minority class will then always have a precision of 100%. Therefore we use a modified f-score in which the precision of a class is 0% if the algorithm never guesses that class. We refer to Birke and Sarkar's (2006) score as *f-score (0/0 = 1)* and to our own score as *f-score (0/0 = 0)*.

Table 5 summarizes our results. *Concrete-Abstract* refers to our own algorithm. *Birke-Sarkar*

[13] Available at http://incubator.apache.org/opennlp/.

[14] Available at http://www.informatics.susx.ac.uk/research/groups/nlp/carroll/morph.html.

refers to the best result reported by Birke and Sarkar (2006), using a form of active learning. *Majority Class* is the simple strategy of always guessing the majority class. *Probability Matching* is the strategy of randomly guessing each class with a probability equal to the size of the class.

Algorithm	Accuracy	F-score (0/0=0)	F-score (0/0=1)
Concrete-Abstract	0.734	0.631	0.639
Birke-Sarkar	*NA*	*NA*	0.649
Majority Class	**0.697**	**0.408**	0.629
Probability Matching	**0.605**	**0.500**	**0.500**

Table 5: The performance with known verbs.

We used a paired t-test to evaluate the statistical significance of the results in Table 5. The numbers are in bold font when the performance of an algorithm is significantly below the performance of Concrete-Abstract. In no case is any score significantly above the performance of Concrete-Abstract, at the 95% confidence level. *NA* indicates scores that were not calculated by Birke and Sarkar (2006).

4.3 Unknown Verbs

For the final experiment, we again used the TroFi Example Base, but with a different experimental setup. Instead of ten-fold cross-validation, we used the twenty-five verbs in Birke and Sarkar (2006) for testing (we call these the *old* verbs) and the other twenty-five verbs (the *new* verbs) for training. The twenty-five old (testing) verbs appear in 1,965 sentences and the twenty-five new (training) verbs appear in 1,772 sentences. For this experiment, we no longer learn a separate logistic regression model for each verb. All of the 1,772 training sentences are used together to learn a single logistic regression model, which is then evaluated on the testing sentences.

Table 6 summarizes our results. Since the testing set is exactly the same as in Section 4.2, we can compare the performance directly with the performance in the preceding section and with Birke and Sarkar's (2006) results.

Again, we used a paired t-test to evaluate the statistical significance of the results in Table 6. The numbers are in bold font when the performance of an algorithm is significantly below the performance of Concrete-Abstract. In no case is any score significantly above the performance of Concrete-Abstract, at the 95% confidence level.

Algorithm	Accuracy	F-score (0/0=0)	F-score (0/0=1)
Concrete-Abstract	0.686	0.673	0.681
Birke-Sarkar	*NA*	*NA*	0.649
Majority Class	0.697	**0.408**	**0.629**
Probability Matching	**0.605**	**0.500**	**0.500**

Table 6: The performance with unknown verbs.

Table 7 shows the coefficients in the logistic regression model that was learned on the training data. The items numbered from 1 to 5 are the five features described in Section 4.2. The sixth item is the constant term in the regression equation. We see that the abstractness of the nouns (excluding proper nouns) has the largest weight in predicting whether the target verb is in class *N*.

	Feature	Coefficient
1	AvgNounAbs	11.4117
2	AvgPropAbs	0.7250
3	AvgVerbAbs	-0.5528
4	AvgAdjAbs	1.1478
5	AvgAdvAbs	-0.2013
6	Intercept	-5.9436

Table 7: The logistic regression coefficients for class *N*.

5 Discussion

It is a strength of our approach that it can classify verbs that it has never seen before, as we see in Section 4.3. The feature vectors in all three experiments are based only on the context; the target adjective or verb is not used in the vectors. This avoids the need for gathering training data on every verb or adjective for which we want to determine whether it is being used metaphorically or literally, since the algorithm is not sensitive to the specific target word.

On the other hand, the performance might improve if the target word were included in the feature vectors. If metaphor is a method for transferring knowledge from concrete domains to abstract domains, then it follows that highly abstract target words will tend to be used literally in most contexts. For instance, the highly abstract verb *epitomize* (with an abstractness rating of 0.85861) is perhaps almost always used in a literal sense. There-

fore it would seem that the abstractness rating of the target word could be a useful clue for determining whether the sense is literal or metaphorical.

We experimented with including the abstractness rating of the target word as a feature, but the impact on performance was not significant for either the adjectives or the verbs. We hypothesize that this may be due to the relatively narrow range in the abstractness of the adjectives and verbs in our data. The abstractness ratings of the adjectives vary from 0.43356 for *dark* to 0.56637 for *hard*. The abstractness ratings of the fifty verbs range from 0.28756 for *plant* to 0.71628 for *lend*, but 80% of the verbs lie in the range from 0.41879 for *fly* to 0.59912 for *rest*. It seems possible that the abstractness rating of the target word would be useful with a dataset in which the target's abstractness varied substantially.

In future work, we would like to gather data for target words with a wider range of abstractness. We expect that such data would show some benefit to including information on the abstractness of the target word in the feature vector.

We also expect that a hybrid of classical word sense disambiguation, such as Birke and Sarkar's (2006) algorithm, with abstractness ratings would perform better than either approach alone. Abstractness may provide a good rough estimate of whether a word usage is literal or metaphorical, but it seems likely that knowledge of the specific target word in question will be required for a highly precise answer. This is another worthwhile topic for future research.

Currently there is no algorithm that identifies what kind of concepts and relations are grafted from the source domain to the target domain by metaphorical inference. The algorithm presented in this paper may be used within a constraints-based model of metaphor (Neuman and Nave, 2009) to address this challenge.

Recently there has been some interest in *visualness*, *picturability*, and *imagability*, the degree to which a word is associated with visual imagery (Deschacht and Moens, 2007). Although Xing et al. (2010) use the term *concreteness* in their work, their research is concerned with predicting the difficulty of queries for image retrieval. It could be argued that Xing et al. should be trying to capture *imagability*, not *concreteness*.

The MRC Psycholinguistic Database (Coltheart, 1981) includes words rated for *imagability*. Our algorithm for rating the abstractness of words (Section 2) could easily be trained with the MRC imagability ratings instead of the abstractness ratings. In future work, it would be interesting to evaluate imagability ratings on the TroFi Example Base. It would also be worthwhile to see whether our algorithm can be adapted for image retrieval (Xing et al., 2010) and image annotation (Deschacht and Moens, 2007).

6 Conclusion

Metaphor is ubiquitous, yet recognizing textual entailment is a challenge when words are used metaphorically. An algorithm for distinguishing metaphorical and literal senses of a word will facilitate correct textual inference, which will improve the many NLP applications that depend on textual inference.

We have introduced a new algorithm for measuring the degree of abstractness of a word. Inspired by research in cognitive linguistics (Lakoff and Johnson, 1980), we hypothesize that the degree of abstractness of the context in which a given word appears is predictive of whether the word is used in a metaphorical or literal sense. This hypothesis is supported by three experiments.

A strength of this approach to the problem of distinguishing metaphorical and literal senses is that it readily generalizes to new words, outside of the training data. We do not claim that abstractness is a complete solution to the problem, but it may be a valuable component in any practical system for processing metaphorical text.

Acknowledgments

Part of the work of Yair Neuman, Dan Assaf, and Yohai Cohen has been supported by a grant from the Israel Ministry of Defense. Thanks to the EMNLP reviewers for their helpful comments.

References

Julia Birke and Anoop Sarkar. 2006. A clustering approach for the nearly unsupervised recognition of nonliteral language. In *Proceedings of the 11th Conference of the European Chapter of the Association for*

Computational Linguistics (EACL 2006), pages 329–336.

Julia Birke and Anoop Sarkar. 2007. Active learning for the identification of nonliteral language. In *Proceedings of the Workshop on Computational Approaches to Figurative Language at HLT/NAACL-07*, pages 21–28.

Stefan Büttcher and Charles Clarke. 2005. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *Proceedings of the 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD.

John Caron. 2001. Experiments with LSA scoring: Optimal rank and basis. In *Proceedings of the SIAM Computational Information Retrieval Workshop*, pages 157–169, Raleigh, NC.

Mark Changizi. 2008. Economically organized hierarchies in WordNet and the Oxford English Dictionary. *Cognitive Systems Research*, 9(3):214–228.

Kenneth Church and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Conference of the Association of Computational Linguistics*, pages 76–83, Vancouver, British Columbia.

Max Coltheart. 1981. The MRC psycholinguistic database. *Quarterly Journal of Experimental Psychology*, 33A(4):497–505.

Marcel Danesi. 2003. Metaphorical "networks" and verbal communication: A semiotic perspective on human discourse. *Sign Systems Studies*, 31:341–363.

Mark Davies. 2009. The 385+ million word Corpus of Contemporary American English (1990–2008+): Design, architecture, and linguistic insights. *International Journal of Corpus Linguistics*, 14(2):159–190.

Koen Deschacht and Marie-Francine Moens. 2007. Text analysis for automatic image annotation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 1000–1007.

William B. Dolan. 1995. Metaphor as an emergent property of machine-readable dictionaries. In *Proceedings of the AAAI 1995 Spring Symposium Series: Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity and Generativity*, pages 27–32.

Dedre Gentner, Brian F. Bowdle, Phillip Wolff, and Consuelo Boronat. 2001. Metaphor is like analogy. In D. Gentner, K. J. Holyoak, and B. N. Kokinov, editors, *The analogical mind: Perspectives from Cognitive Science*, pages 199–253. MIT Press, Cambridge, MA.

Chikara Hashimoto and Daisuke Kawahara. 2009. Compilation of an idiom example database for supervised idiom identification. *Language Resources and Evaluation*, 43(4):355–384.

George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. University Of Chicago Press, Chicago, IL.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Saskia Le Cessie and J.C. Van Houwelingen. 1992. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201.

Alexis Madrigal. 2011. Why are spy researchers building a 'Metaphor Program'? *The Atlantic*, May 25.

James H. Martin. 1992. Computer understanding of conventional metaphoric language. *Cognitive Science*, 16(2):233–270.

Zachary Mason. 2004. CorMet: A computational, corpus-based conventional metaphor extraction system. *Computational Linguistics*, 30(1):23–44.

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

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

Yair Neuman and Ophir Nave. 2009. Metaphor-based meaning excavation. *Information Sciences*, 179:2719–2728.

Malvina Nissim and Katja Markert. 2003. Syntactic features and word similarity for supervised metonymy resolution. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 56–63, Sapporo, Japan.

Reinhard Rapp. 2003. Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the Ninth Machine Translation Summit*, pages 315–322.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco.

Xing Xing, Yi Zhang, and Mei Han. 2010. Query difficulty prediction for contextual image retrieval. In *Advances in Information Retrieval*, volume 5993 of *Lecture Notes in Computer Science*, pages 581–585. Springer.

Syntactic Decision Tree LMs: Random Selection or Intelligent Design?

Denis Filimonov†‡
‡Human Language Technology
Center of Excellence
Johns Hopkins University
den@cs.umd.edu

Mary Harper†
†Department of Computer Science
University of Maryland, College Park
mharper@umd.edu

Abstract

Decision trees have been applied to a variety of NLP tasks, including language modeling, for their ability to handle a variety of attributes and sparse context space. Moreover, forests (collections of decision trees) have been shown to substantially outperform individual decision trees. In this work, we investigate methods for combining trees in a forest, as well as methods for diversifying trees for the task of syntactic language modeling. We show that our tree interpolation technique outperforms the standard method used in the literature, and that, on this particular task, restricting tree contexts in a principled way produces smaller and better forests, with the best achieving an 8% relative reduction in Word Error Rate over an n-gram baseline.

1 Introduction

Language Models (LMs) are an essential part of NLP applications that require selection of the most fluent word sequence among multiple hypotheses. The most prominent applications include Automatic Speech Recognition (ASR) and Machine Translation (MT).

Statistical LMs formulate the problem as the computation of the model's probability to generate the word sequence $w_1, w_2, \ldots, w_m$ (denoted as w_1^m), assuming that higher probability corresponds to more fluent hypotheses. LMs are often represented in the following generative form:

$$p(w_1^m) = \prod_{i=1}^{m} p(w_i|w_1^{i-1})$$

Note the context space for this function, w_1^{i-1} is arbitrarily long, necessitating some independence assumption, which usually consists of reducing the relevant context to $n-1$ immediately preceding tokens:

$$p(w_i|w_1^{i-1}) \approx p(w_i|w_{i-n+1}^{i-1}) \quad (1)$$

These distributions are typically estimated from observed counts of n-grams w_{i-n+1}^{i} in the training data. The context space is still far too large[1]; therefore, the models are recursively smoothed using lower order distributions. For instance, in a widely used n-gram LM, the probabilities are estimated as follows:

$$\tilde{p}(w_i|w_{i-n+1}^{i-1}) = \rho(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1}) \cdot \tilde{p}(w_i|w_{i-n+2}^{i-1}) \quad (2)$$

where ρ is a *discounted* probability[2].

Note that this type of model is a simple Markov chain lacking any notion of syntax. It is widely accepted that languages do have some structure. Moreover, it has been shown that incorporating syntax into a language model can improve its performance (Bangalore, 1996; Heeman, 1998; Chelba and Jelinek, 2000; Filimonov and Harper, 2009). A straightforward way of incorporating syntax into a language model is by assigning a *tag* to each word and modeling them *jointly*; then to obtain the proba-

[1] $O(|V|^{n-1})$ in n-gram model with typical order $n = 3 \ldots 5$, and a vocabulary size of $|V| = 10^4 \ldots 10^6$.

[2] We refer the reader to (Chen and Goodman, 1996) for a survey of the discounting methods for n-gram models.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 691–699,
Edinburgh, Scotland, UK, July 27–31, 2011.

bility of a word sequence, the tags must be marginalized out:

$$p(w_1^m) = \sum_{t_1 \ldots t_m} p(w_1^m t_1^m) = \sum_{t_1 \ldots t_m} \prod_{i=1}^{m} p(w_i t_i | w_1^{i-1} t_1^{i-1})$$

An independence assumption similar to Eq. 1 can be made:

$$p(w_i t_i | w_1^{i-1} t_1^{i-1}) \approx p(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \quad (3)$$

A primary goal of our research is to build strong syntactic language models and provide effective methods for constructing them to the research community. Note that the tags in the context of the joint model in Eq. 3 exacerbate the already sparse problem in Eq. 1, which makes the probability estimation particularly challenging. We utilize decision trees for joint syntactic language models to cluster context because of their strengths (reliance on information theoretic metrics to cluster context in the face of extreme sparsity and the ability to incorporate attributes of different types[3]), and at the same time, unlike log-linear models (Rosenfeld et al., 1994), computationally expensive probability normalization does not have to be postponed until runtime.

In Section 2, we describe the details of the syntactic decision tree LM. Construction of a single-tree model is difficult due to the inevitable greediness of the tree construction process and its tendency to overfit the data. This problem is often addressed by interpolating with lower order decision trees. In Section 3, we point out the inappropriateness of backoff methods borrowed from n-gram models for decision tree LMs and briefly describe a generalized interpolation for such models. The generalized interpolation method allows the addition of any number of trees to the model, and thus raises the question: what is the best way to create diverse decision trees so that their combination results in a stronger model, while at the same time keeping the total number of trees in the model relatively low for computational practicality. In Section 4, we explore and evaluate a variety of methods for creating different trees. To support our findings, we evaluate several of the models on an ASR rescoring task in Section 5. Finally, we discuss our findings in Section 6.

[3]For example, morphological features can be very helpful for modeling highly inflectional languages (Bilmes and Kirchhoff, 2003).

2 Joint Syntactic Decision Tree LM

A decision tree provides us with a clustering function $\Phi(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \rightarrow \{\Phi^1, \ldots, \Phi^N\}$, where N is the number of clusters, and clusters Φ^k are disjoint subsets of the context space. The probability estimation for a joint decision tree model is approximated as follows:

$$p(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \approx p(w_i t_i | \Phi(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1})) \quad (4)$$

In the remainder of this section, we briefly describe the techniques that we use to construct such a decision tree Φ and to estimate the probability distribution for the joint model in Eq. 4.

2.1 Decision Tree Construction

We use recursive partitioning to grow decision trees. In this approach, a number of alternative binary splits of the training data associated with a node are evaluated using some *metric*, the best split is chosen, checked against a *stopping rule* (which aims at preventing overfitting to the training data and usually involves a heldout set), and then the two partitions become the child nodes if the stopping rule does not apply. Then the algorithm proceeds recursively into the newly constructed leaves.

Binary splits are often referred to as *questions* about the context because a binary partition can be represented by a binary function that decides whether an element of context space belongs to one partition or the other. We utilize univariate questions where each question partitions the context on one attribute, e.g., w_{i-2} or t_{i-1}. The questions about words and tags are constructed differently:

- The questions q about the words are in the form $q(x) \equiv w_{i+x} \in S$, where x is an integer between $-n+1$ and -1, and $S \subset V$ is a subset of the word vocabulary V. To construct the set S, we take the set of words S_o *observed* at the offset x in the training data associated with the

current node and split it into two complementary subsets $S \cup \bar{S} = S_o$ using the Exchange algorithm (Martin et al., 1998). Because the algorithm is greedy and depends on the initialization, we construct 4 questions per word position using different random initializations of the Exchange algorithm.

Since we need to account for words that were not observed in the training data, we utilize the structure depicted in Figure 1. To estimate the probability at the backoff node (B in Figure 1), we can either use the probability from its grandparent node A or estimate it using a lower order tree (see Section 3), or combine the two. We have observed no noticeable difference between these methods, which suggests that only a small fraction of probability is estimated from these nodes; therefore, for simplicity, we use the probability estimated at the backoff node's grandparent.

- To create questions about tags we create a hierarchical clustering of all tags in the form of a binary tree. This is done beforehand, using the Minimum Discriminating Information algorithm (Zitouni, 2007) with the entire training data set. In this tree, each leaf is an individual tag and each internal node is associated with the subset of tags that the node dominates. Questions about tags are constructed in the form $q(x, k) \equiv t_{i+x} \in T_k$, where k is a node in the tag tree and T_k is the subset of tags associated with that node. The rationale behind constructing tag questions in this form is that it enables a more efficient decoding algorithm than standard HMM decoding (Filimonov and Harper, 2009).

Questions are evaluated in two steps. First the context attribute x is selected using a metric similar to information gain ratio proposed by (Quinlan, 1986):

$$\mathcal{M} = 1 - \frac{H(w_i) - H(w_i|x)}{H(x)} = 1 - \frac{I(x; w_i)}{H(x)}$$

where x is one of the context attributes, e.g., w_{i-2} or t_{i-1}. Then, among the questions about attribute x, we select the question that maximizes the entropy reduction.

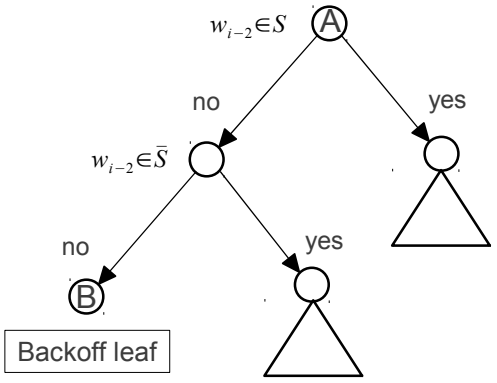


Figure 1: A fragment of the decision tree with a backoff node. $S \cup \bar{S}$ is the set of words observed in the training data at the node A. To account for unseen words, we add the backoff node B.

Instead of dedicating an explicit heldout set for the stopping criterion, we utilize a technique similar to cross validation: the training data set is partitioned into four folds, and the best question is required to reduce entropy on each of the folds.

Note that the tree induction algorithm can also be used to construct trees without tags:

$$p(w_i|w_{i-n+1}^{i-1}) \approx p(w_i|\Phi(w_{i-n+1}^{i-1}))$$

We refer to this model as the *word-tree* model. By comparing syntactic and word-tree models, we are able to separate the effects of decision tree modeling and syntactic information on language modeling by comparing both models to an n-gram baseline.

2.2 In-tree Smoothing

A decision tree offers a hierarchy of clusterings that can be exploited for smoothing. We can interpolate the observed distributions at leaves recursively with their parents, as in (Bahl et al., 1990; Heeman, 1998):

$$\tilde{p}_k(w_i t_i) = \lambda_k p_{ML}(w_i t_i) + (1 - \lambda_k)\tilde{p}_{k'}(w_i t_i) \quad (5)$$

where p_{ML} is the observed distribution at node k and k' is the parent of k. The coefficients λ_k are estimated using an EM algorithm.

We can also combine $p(w_i t_i|\Phi(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}))$ with lower order decision trees, i.e.,

$p(w_i t_i|\Phi(w_{i-n+2}^{i-1} t_{i-n+2}^{i-1}))$, and so on up until $p(w_i t_i)$ which is a one-node tree (essentially a unigram model). Although superficially similar to backoff in n-gram models, lower order decision trees differ substantially from lower order n-gram models and require different interpolation methods. In the next section, we discuss this difference and present a generalized interpolation that is more suitable for combining decision tree models.

3 Interpolation with Backoff Tree Models

In this section, for simplicity of presentation, we focus on the equations for word models, but the same equations apply equally to joint models (Eq. 3) with trivial transformations.

3.1 Backoff Property

Let us rewrite the interpolation Eq. 2 in a more generic way:

$$\tilde{p}(w_i|w_1^{i-1}) = \rho_n(w_i|\Phi_n(w_1^{i-1})) + \gamma(\Phi_n(w_1^{i-1})) \cdot \tilde{p}(w_i|BO_{n-1}(w_1^{i-1})) \quad (6)$$

where, ρ_n is a *discounted* distribution, Φ_n is a clustering function of order n, and $\gamma(\Phi_n(w_1^{i-1}))$ is the backoff weight chosen to normalize the distribution. BO_{n-1} is the *backoff* clustering function of order $n-1$, representing a reduction of context size. In the case of an n-gram model, $\Phi_n(w_1^{i-1})$ is the set of word sequences where the last $n-1$ words are w_{i-n+1}^{i-1}. Similarly, $BO_{n-1}(w_1^{i-1})$ is the set of sequences ending with w_{i-n+2}^{i-1}. In the case of a decision tree model, the same backoff function is typically used, but the clustering function can be arbitrary.

The intuition behind Eq. 6 is that the backoff context $BO_{n-1}(w_1^{i-1})$ allows for a more robust (but less informed) probability estimation than the context cluster $\Phi_n(w_1^{i-1})$. More precisely:

$$\forall_{w_1^{i-1},W} : W \in \Phi_n(w_1^{i-1}) \Rightarrow W \in BO_{n-1}(w_1^{i-1}) \quad (7)$$

that is, every word sequence W that belongs to a context cluster $\Phi_n(w_1^{i-1})$, belongs to the same backoff cluster $BO_{n-1}(w_1^{i-1})$ (hence has the same backoff distribution). For n-gram models, Property 7 trivially holds since $BO_{n-1}(w_1^{i-1})$ and $\Phi_n(w_1^{i-1})$ are defined as sets of sequences ending with w_{i-n+2}^{i-1} and w_{i-n+1}^{i-1}, with the former clearly being a superset of the latter. However, when Φ can be arbitrary, e.g., a decision tree, the property is not necessarily satisfied. Figure 2 illustrates cases when the Property 7 is satisfied (a) and violated (b).

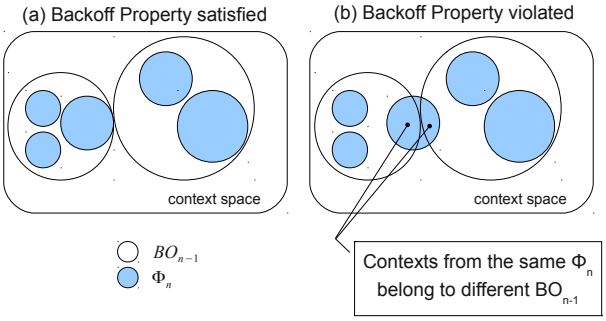


Figure 2: Backoff Property

Let us consider what happens when we have two context sequences W and W' that belong to the same cluster $\Phi_n(W) = \Phi_n(W')$ but different backoff clusters $BO_{n-1}(W) \neq BO_{n-1}(W')$. For example: suppose we have $\Phi(w_{i-2}w_{i-1}) = (\{\text{on}\}, \{\text{may,june}\})$ and two corresponding backoff clusters: $BO' = (\{\text{may}\})$ and $BO'' = (\{\text{june}\})$. Following *on*, the word *may* is likely to be a month rather than a modal verb, although the latter is more frequent and will dominate in BO'. Therefore we have much less faith in $\tilde{p}(w_i|BO')$ than in $\tilde{p}(w_i|BO'')$ and would like a much smaller weight γ assigned to BO'. However this would not be possible in the backoff scheme in Eq. 6, thus we will have to settle on a compromise value of γ, resulting in suboptimal performance.

Hence arbitrary clustering (an advantage of decision trees) leads to a violation of Property 7, which is likely to produce a degradation in performance *if* backoff interpolation Eq. 6 is used.

3.2 Generalized Interpolation

Recursive linear interpolation similar to Jelinek-Mercer smoothing for n-gram models (Jelinek and Mercer, 1980) has been applied to decision tree models:

$$\tilde{p}_n(w_i|w_{i-n+1}^{i-1}) = \lambda_n(\phi_n) \cdot p_n(w_i|\phi_n) + (1 - \lambda_n(\phi_n)) \cdot \tilde{p}_{n-1}(w_i|w_{i-n+2}^{i-1}) \quad (8)$$

where $\phi_n \equiv \Phi_n(w_{i-n+1}^{i-1})$, and $\lambda_n(\phi_n) \in [0, 1]$ are assigned to each cluster and are optimized on a held-out set using EM. $p_n(w_i|\phi_n)$ is the probability distribution at the cluster ϕ_n in the tree of order n. This interpolation method is particularly useful as, unlike count-based discounting methods (e.g., Kneser-Ney), it can be applied to already smoothed distributions p_n.

In (Filimonov and Harper, 2011), we observed that because of the violation of Property 7 in decision tree models, the interpolation method of Eq. 8 is not appropriate for such models. Instead we proposed the following generalized form of linear interpolation:

$$\tilde{p}_n(w_i|w_{i-n+1}^{i-1}) = \frac{\sum_{m=1}^{n} \lambda_m(\phi_m) \cdot p_m(w_i|\phi_m)}{\sum_{m=1}^{n} \lambda_m(\phi_m)} \quad (9)$$

Note that the recursive interpolation of Eq. 8 can be represented in this form with the additional constraint $\sum_{m=1}^{n} \lambda_m(\phi_m) = 1$, which is not required in the generalized interpolation of Eq. 9; thus, the generalized interpolation, albeit having the same number of parameters, has more degrees of freedom. We also showed that the recursive interpolation Eq. 8 is a special case of Eq. 9 that occurs when the Property 7 holds.

4 From Backoff Trees to Forest

Note that, in Eq. 9, individual trees do not have explicit higher-lower order relations, they are treated as a collection of trees, i.e., as a forest. Naturally, to benefit from the forest model, its trees must differ in *some* way. Different trees can be created based on differences in the training data, differences in the tree growing algorithm, or some non-determinism in the way the trees are constructed.

(Xu, 2005) used randomization techniques to produce a large forest of decision trees that were combined as follows:

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{1}{M} \sum_{m=1}^{M} p_m(w_i|w_{i-n+1}^{i-1}) \quad (10)$$

where M is the number of decision trees in the forest (he proposed $M = 100$) and p_m is the m-th tree model[4]. Note that this type of interpolation assumes that each tree model is "equal" a priori and therefore is only appropriate when the tree models are grown in the same way (particularly, using the same order of context). Note that Eq. 10 is a special case of Eq. 9 when all parameters λ are equal.

(Xu, 2005) showed that, although each individual tree is a fairly weak model, their combination outperforms the n-gram baseline substantially. However, we find this approach impractical for online application of any sizable model: In our experiments, fourgram trees have approximately 1.8 million leaves and the tree structure itself (without probabilities) occupies nearly 200MB of disk space after compression. It would be infeasible to apply a model consisting of more than a handful of such trees without distributed computing of some sort. Therefore, we pose the following question: If we can afford to have only a handful of trees in the model, what would be best approach to construct those trees?

In the remainder of this section, we will describe the experimental setup, discuss and evaluate different ways of building decision tree forests for language modeling, and compare combination methods based on Eq. 9 and Eq. 10 (when Eq. 10 is applicable).

4.1 Experimental Setup

To train our models we use 35M words of WSJ 94-96 from LDC2008T13. The text was converted into speech-like form, namely numbers and abbreviations were verbalized, text was downcased, punctuation was removed, and contractions and possessives were joined with the previous word (i.e., *they 'll* becomes *they'll*). For the syntactic modeling, we used tags comprised of the POS tags of the word and it's head. Parsing of the text for tag extraction occurred after verbalization of numbers and abbreviations but

[4]Note that (Xu, 2005) used lower order models to estimate p_m.

before any further processing; we used a latent variable PCFG parser as in (Huang and Harper, 2009). For reference, we include an n-gram model with modified interpolated KN discounting. All models use the same vocabulary of approximately 50k words.

Perplexity numbers reported in Tables 1, 2, 3, and 4 are computed on WSJ section 23 (tokenized in the same way)[5].

In Table 1, we show results reported in (Filimonov and Harper, 2011), which we use as the baseline for further experiments. We constructed two sets of decision trees (a joint syntactic model and a word-tree model) as described in Section 2. Each set was comprised of a fourgram tree with backoff trigram, bigram, and unigram trees. We combined these trees using either Eq. 8 or Eq. 9. The λ parameters in Eq. 8 were estimated using EM by maximizing likelihood of a heldout set (we utilized 4-way cross-validation); whereas, the parameters in Eq. 9 were estimated using L-BFGS because the denominator in Eq. 9 makes the maximization step problematic.

4.2 Random Forest

(Xu, 2005) evaluated a variety of randomization techniques that can be used to build trees. He used a word-only model, with questions constructed using the Exchange algorithm, similar to our model. He tried two methods of randomization: selecting the positions in the history for question construction by a Bernoulli trials[6], and random initialization of the Exchange algorithm. He found that when the Exchange algorithm was initialized randomly, the Bernoulli trial parameter did not matter; however, when the Exchange algorithm was initialized deterministically; lower values for the Bernoulli trial parameter r yielded better overall forest performance. We implemented a similar method, namely, initializing the Exchange algorithm randomly and using $r = 0.1$ for Bernoulli trials[7].

There is a key difference between the two randomization methods. Since we do not have an a priori preference for choosing initializations for the Exchange algorithm, by using random initializations it is *hoped* that due to the greedy nature of the algorithm, the constructed trees, while being "undegraded,"[8] will be sufficiently different so that their combination improves over an individual tree. By introducing Bernoulli trials, on the other hand, there is a choice to purposely *degrade* the quality of individual trees in the hope that additional diversity would enable their combination to compensate for the loss of quality in individual trees.

Another way of introducing randomness to the tree construction without apparent degradation of individual tree quality is through varying the data, e.g., using different folds of the training data (see Section 2.1).

Let us take a closer look at the effect of different types of randomization on individual trees and their combinations. In the first set of experiments, we compare the performance of a single undegraded fourgram tree[9] with forests of fourgram trees grown randomly with Bernoulli trials. Having only same-order trees in a forest allows us to apply interpolation of Eq. 10 (used in (Xu, 2005)) and compare with the interpolation method presented in Eq. 9. By comparing forests of different sizes with the baseline from Table 1, we are able to evaluate the effect of randomization in decision tree growing and assess the importance of the lower order trees.

The results are shown in Table 2. Note that, while an undegraded syntactic tree is better than the word tree, the situation is reversed when the trees are grown randomly. This can be explained by the fact that the joint model has a much higher dimensionality of the context space, and therefore is much more sensitive to the clustering method.

As we increase the number of random trees in the forest, the perplexity decreases as expected, with the interpolation method of Eq. 9 showing improvement of a few percentile points over Eq. 10. Note that in the case of the word-tree model, it takes 4 random decision trees to reach the performance of a single undegraded tree, while in the joint model, even

[5]This section was not used for training the parser or for the LM training.

[6]In this method, positions in the history are ignored with probability $1 - r$, where r is the Bernoulli trials parameter.

[7]Note that because in the joint model, the question about tags are deterministic, we use a lower value of r than (Xu, 2005) to increase randomness.

[8]Here and henceforth, by "undegraded" we mean "according to the algorithm described in Section 2."

[9]Since each tree has a smooth distribution based on Eq. 5, lower order trees are not strictly required.

		Eq. 8		Eq. 9 (generalized)	
order	n-gram	word-tree	syntactic	word-tree	syntactic
2-gram	261.0	257.8	214.3	258.1	214.6
3-gram	174.3 (33.2%)	168.7 (34.6%)	156.8 (26.8%)	168.4 (34.8%)	155.3 (27.6%)
4-gram	161.7 (7.2%)	164.0 (2.8%)	156.5 (0.2%)	155.7 (7.5%)	147.1 (5.3%)

Table 1: Perplexity results on PTB WSJ section 23. Percentage numbers in parentheses denote the reduction of perplexity relative to the lower order model of the same type.

	word-tree		syntactic	
	Eq. 10	Eq. 9	Eq. 10	Eq. 9
1 × undgr	204.9		189.1	
1 × rnd	250.2		289.9	
2 × rnd	229.5	221.5	244.6	240.9
3 × rnd	227.5	214.5	226.2	220.0
4 × rnd	219.5	205.0	219.5	212.2
5 × rnd	200.9	184.1	216.5	209.0
baseline	N/A	155.7	N/A	147.1

Table 2: Perplexity numbers obtained using fourgram trees only. Note that "undgr" and "rnd" denote undegraded and randomly grown trees with Bernoulli trials, respectively, and the number indicates the number of trees in the forest. Also "baseline" refers to the fourgram models with lower order trees (from Table 1, Eq. 9).

	word-tree		syntactic	
# trees	Exchng.	+data	Exchng.	+data
1	204.9		189.1	
2	185.9	186.5	174.5	173.7
3	179.5	179.9	168.8	167.2
4	176.2	176.4	165.1	164.0
5	173.7	172.0	163.0	162.0
baseline	155.7		147.1	

Table 3: Perplexity numbers obtained using fourgram trees produced using random initialization of the Exchange algorithm (Exchng. columns) and, additionally, variations in training data folds (+data columns). Note that "baseline" refers to the fourgram models with lower order trees (from Table 1). All models use the interpolation method of Eq. 9.

5 trees are much worse than a single decision tree constructed without randomization. Finally, compare the performance of single undegraded fourgram trees in Table 2 with fourgram models in Table 1, which are constructed with lower order trees: both word-tree and joint models in Table 1 have over 20% lower perplexity compared to the corresponding models consisting of a single fourgram tree.

In Table 3, we evaluate forests of fourgram trees produced using randomizations without degrading the tree construction algorithm. That is, we use random initializations of the Exchange algorithm and, additionally, variations in the training data fold. All forests in this table use the interpolation method of Eq. 9. Note that, while these perplexity numbers are substantially better than trees produced with Bernoulli trials in Table 2, they are still significantly worse than the baseline model from Table 1.

These results suggest that, while it is beneficial to combine *different* decision trees, we should introduce differences to the tree construction process without degrading the trees when introducing randomness, especially for joint models. In addition, lower order trees seem to play an important role for high quality model combination.

4.3 Context-Restricted Forest

As we have mentioned above, combining higher and lower order decision trees produces much better results. A lower order decision tree is grown from a lower order context space, i.e., the context space where we purposely ignore some attributes. Note that in this case, rather than *randomly* ignoring contexts via Bernoulli trials at every node in the decision tree, we discard some context attributes upfront in a principled manner (i.e., most distant context) and then grow the decision tree without degradation.

Since the joint model, having more context attributes, affords a larger variety of different contexts, we use this model in the remaining experiments.

In Table 4, we present the perplexity numbers for our standard model with additional trees. We denote context-restricted trees by their Markovian or-

Model	size	PPL
1w1t + 2w2t + 3w3t + 4w4t (*)	294MB	147.1
(*) + 4w3t + 3w2t	579MB	143.5
(*) + 4w3t + 3w4t	587MB	144.9
(*) + 4w3t + 3w4t + 3w2t + 2w3t	699MB	**140.7**
(*) + 1 × bernoulli-rnd	464MB	149.7
(*) + 2 × bernoulli-rnd	632MB	150.4
(*) + 3 × bernoulli-rnd	804MB	151.1
(*) + 1 × data-rnd	484MB	147.0
(*) + 2 × data-rnd	673MB	145.0
(*) + 3 × data-rnd	864MB	145.2

Table 4: Perplexity results using the standard syntactic model with additional trees. "bernoulli-rnd" and "data-rnd" indicate fourgram trees randomized using Bernoulli trials and varying training data, respectively. The second column shows the combined size of decision trees in the forest.

ders (words w and tags t independently), so 3w2t indicates a decision tree implementing the probability function: $p(w_i t_i | w_{i-1} w_{i-2} t_{i-1})$. The fourgram joint model presented in Table 1 has four trees and is labeled with the formula "1w1t + 2w2t + 3w3t + 4w4t" in Table 4. The randomly grown trees (denoted "bernoulli-rnd") are grown utilizing the full context 4w4t using the methods described in Section 4.2. All models utilize the generalized interpolation method described in Section 3.2.

As can be seen in Table 4, adding undegraded trees consistently improves the performance of an already strong baseline, while adding random trees only increases the perplexity because their quality is worse than undegraded trees'. Trees produced by data randomization (denoted "data-rnd") also improve the performance of the model; however, the improvement is not greater than that of additional lower order trees, which are considerably smaller in size.

5 ASR Rescoring Results

In order to verify that the improvements in perplexity that we observe in Tables 1 and 4 are sufficient for an impact on a task, we measure Word Error Rate (WER) of our models on an Automatic Speech Recognition (ASR) rescoring task using the Wall Street Journal corpus (WSJ) for evaluation. The test set consists of 4,088 utterances of WSJ0. We optimized the weights for the combination of acoustic and language model scores on a separate development set comprised of 1,243 utterances from Hub2 5k closed vocabulary and the WSJ1 5k open vocabulary sets.

Model	PPL	WER
n-gram	161.7	7.81%
1w1t + 2w2t + 3w3t + 4w4t (Eq.8)	156.5	7.57%
1w1t + 2w2t + 3w3t + 4w4t (*)	147.1	7.32%
(*) + 4w3t + 3w4t + 3w2t + 2w3t	140.7	**7.20%**

Table 5: Perplexity and WER results. Note that the last two rows are syntactic models using the interpolation method of Eq. 9.

The ASR system used to produce lattices is based on the 2007 IBM Speech transcription system for the GALE Distillation Go/No-go Evaluation (Chen et al., 2006). The acoustic models are state-of-the-art discriminatively trained models which are trained on Broadcast News (BN) Hub4 acoustic training data. Lattices were produced using a trigram LM trained on the same data as the models we evaluate, then 1,000 best unique hypotheses were extracted from the lattices. WER of the 1-best hypothesis on the test set is 8.07% and the oracle WER is 3.54%.

In Table 5, we present WER results along with the corresponding perplexity numbers from Tables 1 and 4 for our lowest perplexity syntactic model, as well as the baselines (modified KN n-gram model and standard decision tree models using interpolation methods of Eq. 8 and Eq. 9). The interpolation method of Eq. 9 substantially improves performance over the interpolation method of Eq. 8, reducing WER by 0.25% absolute ($p < 10^{-5}$). Adding four trees utilizing context restricted in different ways further reduces WER by 0.12%, which is also a statistically significant ($p < 0.025$) improvement over the baseline models labeled (*). Altogether, the improvements over the n-gram baseline add up to 0.61% absolute (8% relative) WER reduction.

6 Conclusion

In this paper, we investigate various aspects of combining multiple decision trees in a single language model. We observe that the generalized interpola-

tion (Eq. 9) for decision tree models proposed in (Filimonov and Harper, 2011) is in fact a forest interpolation method rather than a backoff interpolation because, in Eq. 9, models do not have explicit higher-lower order relation as they do in backoff interpolation (Eq. 6). Thus, in this paper we investigate the question of how to construct decision trees so that their combination results in improved performance (under the assumption that computational tractability allows only a handful of decision trees in a forest). We compare various techniques for producing forests of trees and observe that methods that diversify trees by introducing random degradation of the tree construction algorithm perform more poorly (especially with joint models) than methods in which the trees are constructed without degradation and with variability being introduced via parameters that are inherently arbitrary (e.g., training data fold differences or initializations of greedy search algorithms). Additionally, we observe that simply restricting the context used to construct trees in different ways, not only produces smaller trees (because of the context reduction), but the resulting variations in trees also produce forests that are *at least* as good as forests of larger trees.

7 Acknowledgments

We would like to thank Ariya Rastrow for providing word lattices for the ASR rescoring experiments.

References

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. 1990. A tree-based statistical language model for natural language speech recognition. *Readings in speech recognition*, pages 507–514.

Srinivas Bangalore. 1996. 'Almost parsing' technique for language modeling. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 1173–1176.

Jeff Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of HLT/NAACL*, pages 4–6.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling for speech recognition. *CoRR*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318.

S. Chen, B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig. 2006. Advances in speech transcription at IBM under the DARPA EARS program. *IEEE Transactions on Audio, Speech and Language Processing*, pages 1596–1608.

Denis Filimonov and Mary Harper. 2009. A joint language model with fine-grain syntactic tags. In *Proceedings of the EMNLP 2009*.

Denis Filimonov and Mary Harper. 2011. Generalized interpolation in decision tree LM. In *Proceedings of the 49st Annual Meeting of the Association for Computational Linguistics*.

Peter Heeman. 1998. POS tagging versus classes in language modeling. In *Sixth Workshop on Very Large Corpora*.

Zhongqiang Huang and Mary Harper. 2009. Self-Training PCFG grammars with latent annotations across languages. In *Proceedings of the EMNLP 2009*.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397.

Sven Martin, Jorg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. In *Speech Communication*, pages 1253–1256.

J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1(1):81–106.

Ronald Rosenfeld, Jaime Carbonell, and Alexander Rudnicky. 1994. Adaptive statistical language modeling: A maximum entropy approach. Technical report.

Peng Xu. 2005. *Random Forests and Data Sparseness Problem in Language Modeling*. Ph.D. thesis, Baltimore, Maryland, April.

Imed Zitouni. 2007. Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. *Computer Speech & Language*, 21(1):88–104.

The Imagination of Crowds: Conversational AAC Language Modeling using Crowdsourcing and Large Data Sources

Keith Vertanen
Department of Computer Science
Princeton University
vertanen@princeton.edu

Per Ola Kristensson
School of Computer Science
University of St Andrews
pok@st-andrews.ac.uk

Abstract

Augmented and alternative communication (AAC) devices enable users with certain communication disabilities to participate in everyday conversations. Such devices often rely on statistical language models to improve text entry by offering word predictions. These predictions can be improved if the language model is trained on data that closely reflects the style of the users' intended communications. Unfortunately, there is no large dataset consisting of genuine AAC messages. In this paper we demonstrate how we can crowdsource the creation of a large set of fictional AAC messages. We show that these messages model conversational AAC better than the currently used datasets based on telephone conversations or newswire text. We leverage our crowdsourced messages to intelligently select sentences from much larger sets of Twitter, blog and Usenet data. Compared to a model trained only on telephone transcripts, our best performing model reduced perplexity on three test sets of AAC-like communications by 60–82% relative. This translated to a potential keystroke savings in a predictive keyboard interface of 5–11%.

1 Introduction

Users with certain communication disabilities rely on augmented and alternative communication (AAC) devices to take part in everyday conversations. Often these devices consist of a predictive text input method coupled with text-to-speech output. Unfortunately, the text entry rates provided by AAC devices are typically low, between 0.5 and 16 words-per-minute (Trnka et al., 2009).

As a consequence, researchers have made numerous efforts to increase AAC text entry rates by employing a variety of improved language modeling techniques. Examples of approaches include adapting the language model to recently used words (Wandmacher et al., 2008; Trnka, 2008), using syntactic information (Hunnicutt, 1989; Garay-Vitoria and González-Abascal, 1997), using semantic information (Wandmacher and Antoine, 2007; Li and Hirst, 2005), and modeling topics (Lesher and Rinkus, 2002; Trnka et al., 2006). For a recent survey, see Garay-Vitoria and Abascal (2006).

While such language model improvement techniques are undoubtedly helpful, certainly they can all benefit from starting with a long-span language model trained on large amounts of closely matched data. For AAC devices this means closely modeling everyday face-to-face communications. However, a long-standing problem in the field is the lack of good data sources that adequately model such AAC communications. Due to privacy-reasons and other ethical concerns, there is no large dataset consisting of genuine AAC messages. Therefore, previous research has used transcripts of telephone conversations or newswire text. However, these data sources are unlikely to be an ideal basis for AAC language models.

In this paper we show that it is possible to significantly improve conversational AAC language modeling by first crowdsourcing the creation of a fictional collection of AAC messages on the Amazon Mechanical Turk microtask market. Using a care-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 700–711,
Edinburgh, Scotland, UK, July 27–31, 2011.

fully designed microtask we collected 5890 messages from 298 unique workers. As we will see, word-for-word these fictional AAC messages are better at predicting AAC test sets than a wide-range of other text sources. Further, we demonstrate that Twitter, blog and Usenet data outperform telephone transcripts or newswire text.

While our crowdsourced AAC data is better than other text sources, it is too small to train high-quality long-span language models. We therefore investigate how to use our crowdsourced collection to intelligently select AAC-like sentences from Twitter, blog and Usenet data. We compare a variety of different techniques for doing this intelligent selection. We find that the best selection technique is the recently proposed cross-entropy difference method (Moore and Lewis, 2010). Using this method, we build a compact and well-performing mixture model from the Twitter, blog and Usenet sentences most similar to our crowdsourced data.

We evaluate our mixture model on four different test sets. On the three most AAC-like test sets, we found substantial reductions in not only perplexity but also in potential keystroke savings when used in a predictive keyboard interface. Finally, to aid other AAC researchers, we have publicly released our crowdsourced AAC collection, word lists and best-performing language models[1].

2 Crowdsourcing AAC-like Messages

As we mentioned in the introduction, there are unfortunately no publicly available sources of genuine conversational AAC messages. We conjectured we could create surrogate data by asking workers on Amazon Mechanical Turk to imagine they were a user of an AAC device and having them invent things they might want to say. While crowdsourcing is commonly used for simple human computation tasks, such as labeling images and transcribing audio, it is an open research question whether we can leverage workers' creativity to invent plausible and useful AAC-like messages. In this section, we describe our carefully constructed microtask and compare how well our collected messages correspond to communications from actual AAC users.

[1] http://www.aactext.org/imagine/

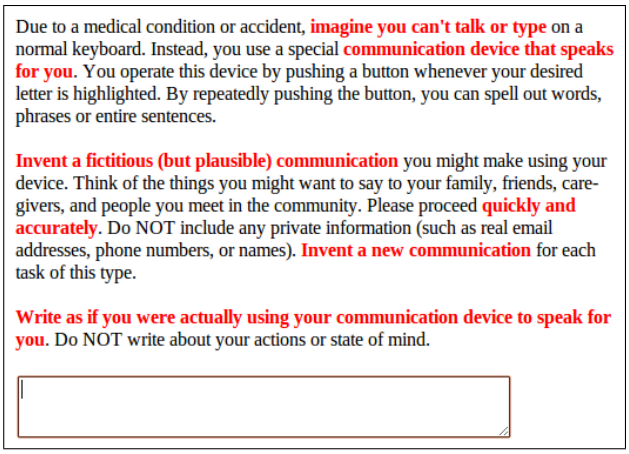


Figure 1: The interface for HITs of type 1 in our crowdsourced data collection.

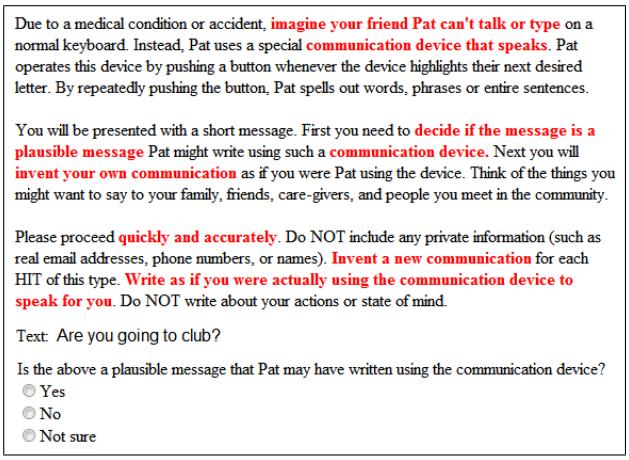


Figure 2: The interface for HITs of type 2 in our crowdsourced data collection.

2.1 Collection Tasks

To collect our data, we used two different types of human intelligence tasks (HITs). In type 1, the workers were told to imagine that due to an accident or medical condition they had to use a communication device to speak for them. Workers were asked to invent a plausible communication. Workers were prevented from pasting text. After several pilot experiments, we arrived at the instructions shown in figure 1.

In type 2, a worker first judged the plausibility of a communication written by a previous worker (figure 2). After judging, the worker was asked to "invent a completely new communication" as if the worker was the AAC user. Workers were prevented from pasting text or typing the identical text as the one just judged. The same communication

was judged by three separate workers. In this work we did not make use of these judgments.

2.2 Data Cleaning

While most workers produced plausible and often creative communications, some workers entered obvious garbage. These workers were identified by a quick visual scan of the submitted communications. We rejected the work of 9% of the workers in type 1 and 4% of the workers in type 2. After removing these workers, we had 2481 communications from type 1 and 4440 communications from type 2.

After combining the data from all accepted HITs, we conducted further semi-automatic data cleaning. We first manually reviewed communications sorted by worker. We removed workers whose text was non-fluent English or not plausible (e.g. some workers entered news headlines or proverbs). Identical communications from the same worker were removed. We removed communications with an out-of-vocabulary (OOV) rate of over 20% with respect to a large word list of 330K words obtained from human-edited dictionaries[2]. We also removed communications that were all in upper case, contained common texting abbreviations (e.g. "plz", "ru", "2day"), communications over 80 characters, and communications with excessive letter repetitions (e.g. "yippeeee"). After cleaning, we had 5890 messages from 298 unique workers.

2.3 Results

Tables 1 and 2 show some example communications obtained in each HIT type. Sometimes, but not always, type 2 resulted in the worker writing a similar communication as the one judged. This is a mixed blessing. While it may reduce the diversity of communications, we found that workers were more eager to accept HITs of type 2. The average HIT completion time was also shorter, 24 seconds in type 2 versus 36 seconds in type 1. While we initially paid $0.04/HIT for both types, we found in subsequent rounds that we could pay $0.02/HIT for type 2. We also had to reject less work in type 2 and qualitatively found the communications to be more AAC-like. Since workers had to imagine themselves in a very unfamiliar situation, it appears that providing a concrete example was helpful to workers.

[2]We combined Wiktionary, Webster's dictionary provided by Project Gutenberg, the CMU pronouncing dictionary and GNU aspell.

Is the dog friendly?
Can I have some water please?
I need to start making a shopping list soon.
What I would really like right now is a plate of fruit.
Who will drive me to the doctor's office tomorrow?

Table 1: Example communications from type 1.

Can you bring my slippers?
I am cold, is there another blanket.
How did Pam take the news?
Bring the fuzzy slippers here.
Did you have breakfast?
why are you so late?
I am pretty hungry, can we go eat?
I had bacon eggs and hashbrowns for breakfast.

Table 2: Example communications from type 2. The text in bold is the message workers judged. It is followed in plain text by the workers' new messages.

3 Comparison of Training Sources

In this section, we compare the predictive performance of language models trained on our Turk AAC data with models trained on other text sources. We use the following training sets:

- NEWS – Newspaper articles from the CSR-III (Graff et al., 1995) and Gigaword corpora (Graff, 2003). 60M sentences, 1323M words.
- WIKIPEDIA – Current articles and discussion threads from a snapshot of Wikipedia (January 3, 2008). 24M sentences, 452M words.
- USENET – Messages from a Usenet corpus (Shaoul and Westbury, 2009). 123M sentences, 1847M words.
- SWITCHBOARD – Transcripts of 2217 telephone conversations from the Switchboard corpus (Godfrey et al., 1992). Due to its conversational style, this corpus has been popular for AAC language modeling (Lesher and Rinkus, 2002; Trnka et al., 2009). 0.2M sentences, 2.6M words.
- BLOG – Blog posts from the ICWSM corpus (Burton et al., 2009). 25M sentences, 387M words.

- TWITTER – We collected Twitter messages via the streaming API between December 2010 and March 2011. We used the free Twitter stream which provides access to 5% of all tweets. Twitter may be particularly well suited for modeling AAC communications as tweets are short typed messages that are often informal person-to-person communications. Twitter has previously been proposed as a candidate for modeling conversations, see for example Ritter et al. (2010). 7M sentences, 55M words.
- TURKTRAIN – Communications from 80% of the workers in our crowdsourced collection. 4981 sentences, 24860 words.

WIKIPEDIA, USENET, BLOG and TWITTER all consisted of raw text that required significant filtering to eliminate garbage, spam, repeated messages, XML tags, non-English text, etc. Given the large amount of data available, our approach was to throw away any text that did not appear to be a sensible English sentence. For example, we eliminated any sentence having a large number of words not in our 330K word list.

3.1 Test Sets

We evaluated our models on the following test sets:

- COMM – Sentences written in response to hypothetical communication situations collected by Venkatagiri (1999). We removed nine sentences containing numbers. This set is used throughout the paper. 251 sentences, 1789 words.
- SPECIALISTS – Context specific phrases suggested by AAC specialists[3]. This set is used throughout the paper. 952 sentences, 3842 words.
- TURKDEV – Communications from 10% of the workers in our crowdsourced collection (disjoint from TURKTRAIN and TURKTEST). This set will be used for initial evaluations and also to tune our models. 551 sentences, 2916 words.
- TURKTEST – Communications from 10% of the workers in our crowdsourced collection (disjoint from TURKTRAIN and TURKDEV). This set is used only in the final evaluation section. 563 sentences, 2721 words.

[3] http://aac.unl.edu/vocabulary.html

Test set	Sentence
COMM	I love your new haircut.
COMM	How many children do you have?
SPECIALISTS	Are you sure you don't mind?
SPECIALISTS	I'll keep an eye on that for you
SWITCHTEST	yeah he's a good actor though
SWITCHTEST	what did she have like

Table 3: Examples from three of our test sets.

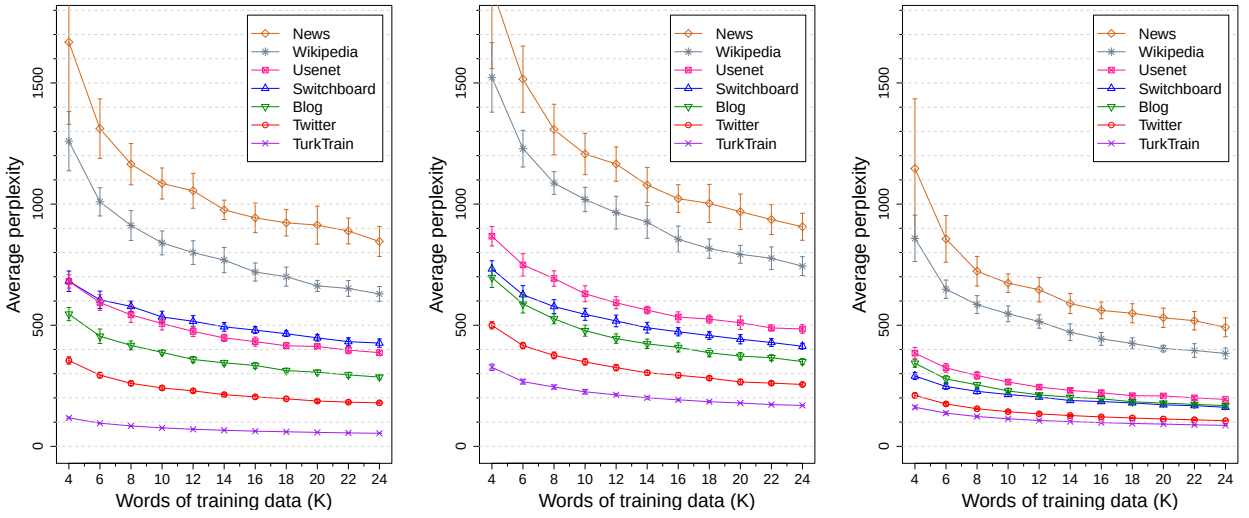
- SWITCHTEST – Transcripts of three Switchboard conversations (disjoint from the SWITCHBOARD training set). This is the same set used in Trnka et al. (2009). We dropped one sentence containing a dash. This set is only used in the final evaluation section. 59 sentences, 508 words.

TURKDEV and TURKTEST contain text similar to table 1 and 2. Table 3 shows some examples from the other three test sets. Sentences in COMM tended to be richer in vocabulary and subject matter than those in SPECIALISTS. The SPECIALISTS sentences tended to be general phrases that avoided mentioning specific situations, proper names, etc. Sentences in SWITCHTEST exhibited phenomena typical of human-to-human voice conversations (filler words, backchannels, interruptions, etc).

3.2 Language Model Training

All language models were trained using the SRILM toolkit (Stolcke, 2002). All models used interpolated modified Kneser-Ney smoothing (Kneser and Ney, 1995; Chen and Goodman, 1998). In this section, we trained 3-gram language models with no count-cutoffs. All text was converted to lowercase and we removed punctuation except for apostrophes. We believe punctuation would likely slow down a user's conversation for only a small potential advantage (e.g. improving text-to-speech prosody).

All models used a vocabulary of 63K words including an unknown word. We obtained our vocabulary by taking all words occurring in TURKTRAIN and all words occurring four or more times in the TWITTER training set. We restricted our vocabulary to words from our large list of 330K words. This restriction prevented the inclusion of common misspellings prevalent in many of our training sets. Our 63K vocabulary resulted in low OOV



(a) TURKDEV test set (b) COMM test set (c) SPECIALISTS test set

Figure 3: Perplexity of language models trained on the same amount of data from different sources. The perplexity is the average of 20 models trained on random subsets of the training data (one standard deviation error bars).

rates for all test sets: COMM 0%, SPECIALISTS 0.05%, TURKDEV 0.1%, TURKTEST 0.07%, and SWITCHTEST 0.8%.

3.3 Small Training Size Experiment

We trained language models on each dataset, varying the number of training words from 4K to 24K (the limit of the TURKTRAIN set). For each dataset and training amount, we built 20 different models by choosing sentences from the full training set at random. We computed the mean and standard deviation of the per-word perplexity of the set of 20 models.

As shown in figure 3, word-for-word the TURKTRAIN data was superior for our three most AAC-like test sets. Thus it appears our crowdsourcing procedure was successful at generating AAC-like data. TWITTER was consistently the second best. BLOG, USENET and SWITCHBOARD also performed well.

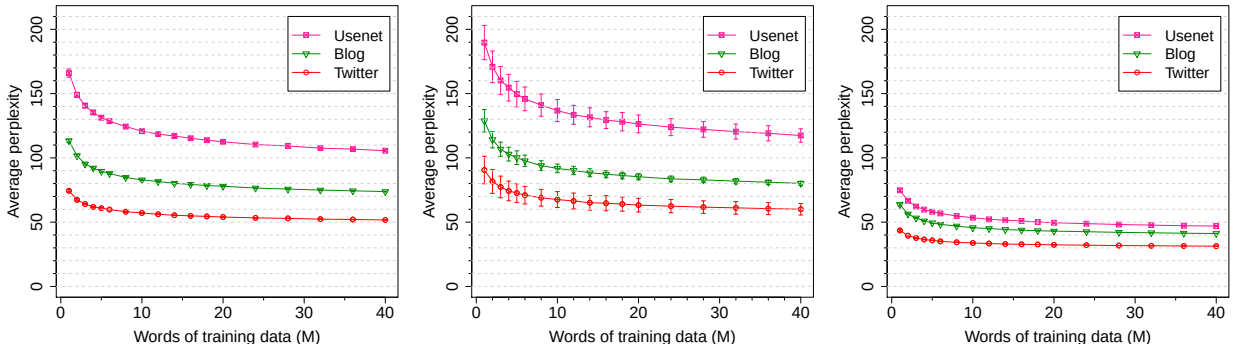
3.4 Large Training Size Experiment

The previous experiment used a small amount of training data. We selected the best three datasets having tens of millions of words of training data: USENET, BLOG, and TWITTER. As in the previous experiment, we computed the mean and standard deviation of the per-word perplexity of a set of 20 models. Increasing the amount of training data substantially reduced perplexity compared to our small TURKTRAIN collection (figure 4). Tweets were clearly well suited for modeling AAC-like text as 3M words of TWITTER data was better than 40M words of BLOG data.

3.5 Comparison with Real AAC Data

Beukelman et al. (1984) analyzed the communications made by five nonspeaking adults over 14 days. All users were experienced using a tape-typewriter AAC device. Beukelman gives a ranked list of the top 500 words, the frequency of the top 20 words, and statistics calculated on the communications.

For the top 10 words in Beukelman's AAC user data, we computed the probability of each word in our various datasets (figure 5). As shown, some words such as "to" and "a" occur with similar frequency across all datasets. Some words such as "the" are overrepresented in data such as news text. Other words such as "I" and "you" are much more variable. Our Turk data has the closest matching frequency for the most popular word "I". Interestingly, our Turk data shows a much higher probability for "you" than the AAC data. We believe this resulted from the situation we asked workers to imagine (i.e. communicating via a letter-at-a-time scanning interface). Workers presumed in such a situation they would need to ask others to do many tasks. We observed many requests in the data such as "Can



(a) TURKDEV test set (b) COMM test set (c) SPECIALISTS test set

Figure 4: Perplexity of language models trained on increasing amounts of data from three different training sources. Results on the TURKDEV, COMM and SPECIALISTS test sets.

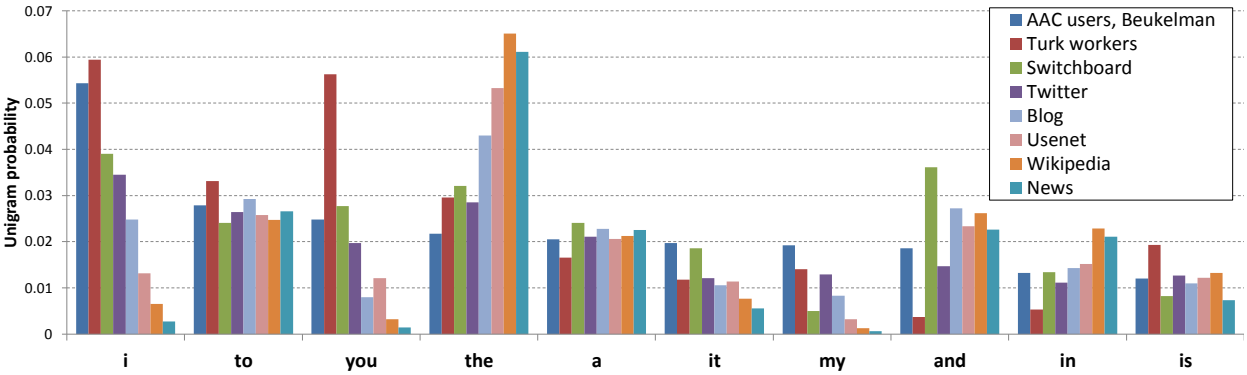


Figure 5: The unigram probabilities of the top 10 words reported by Beukelman et al. (1984).

you change my sheets?" and "Can you walk the dog for me?"

Beukelman reports 33% of all communications could be made using only the top 500 words. The same 500 words allowed writing of 34% of our Turk communications. Other datasets exhibited much lower percentages. Note that this is at least partially due to the longer sentences present in some datasets. Unfortunately, Beukelman does not report the average communication length. Our Turk communications were 5.0 words on average. The next shortest dataset was TWITTER with 7.5 words per communication. Despite their short average length, only 10% of Tweets could be written using the top 500 words.

Beukelman reports that 80% of words in the AAC users' communications were in the top 500 words. 81% of the words in our crowdsourced data were in this word list. For comparison, only 65% of words in our TWITTER data were in the 500 word vocabulary. While our TURKTRAIN set contains only 2141 unique words, this may in fact be good since it has been argued that rare words have received too much attention in AAC (Baker et al., 2000).

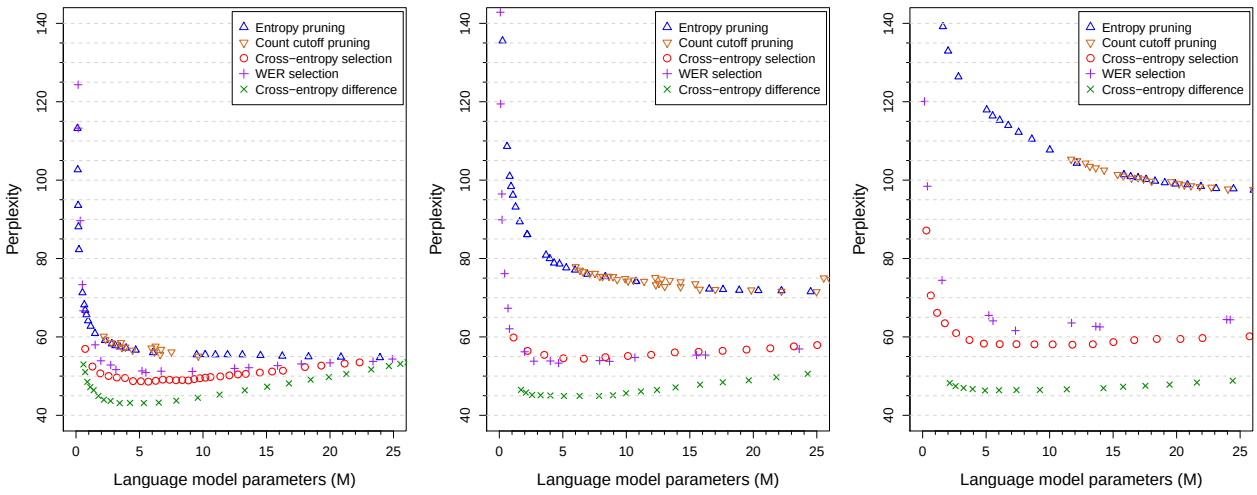
4 Using Large Datasets Effectively

In the previous section, we found our crowdsourced data was good at predicting AAC-like test sets. However, in order to build a good long-span language model, we would require millions of such communications. Crowdsourcing such a large collection would be prohibitively expensive. Therefore, we instead investigated how to use our crowdsourced data to intelligently select AAC-like data from other large datasets. For large datasets, we used TWITTER, BLOG and USENET as they were both large and well-matched to AAC data.

4.1 Selecting AAC-like Data

For each training sentence, we calculated three values:

- WER – The minimum word error rate between the training sentence and one of the crowdsourced



(a) TWITTER (b) BLOG (c) USENET

Figure 6: Perplexity on TURKDEV using different data selection and pruning techniques.

communications. This is the minimum number of words that must be inserted, substituted or deleted to transform the training sentence into a TURKTRAIN sentence divided by the number of words in the TURKTRAIN sentence. For example, the training sentence "I didn't sleep well Monday night either" was given a WER of 0.33 because two word-changes transformed it into a message written by a worker: "I didn't sleep well last night".

- Cross-entropy, in-domain – The average per-word cross-entropy of the training sentence under a 3-gram model trained on TURKTRAIN.
- Cross-entropy, background – The average per-word cross-entropy of the training sentence under a 3-gram model trained on a random portion of the training set. The random portion was the same size as TURKTRAIN.

We used these values to limit training to only AAC-like sentences. We tried three different selection methods. In *WER selection*, only sentences below a threshold on the word error rate were kept in the training data. This tends to find variants of existing communications in our Turk collection.

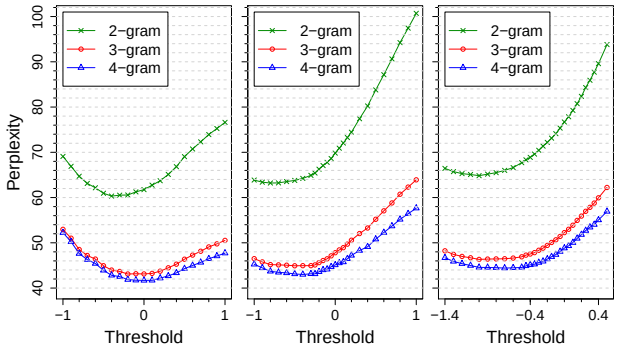
In *cross-entropy selection*, we used only sentences below a threshold on the per-word cross-entropy with respect to a TURKTRAIN language model. This is equivalent to placing a threshold on the perplexity. Previously this technique has been used to improve language models based on web data (Bulyko et al., 2007; Gao et al., 2002) and to construct domain-specific models (Lin et al., 1997).

In *cross-entropy difference selection*, a sentence's score is the in-domain cross-entropy minus the background cross-entropy (Moore and Lewis, 2010). This technique has been used to supplement European parliamentary text (48M words) with newswire data (3.4B words) (Moore and Lewis, 2010). We were curious how this technique would work given our much smaller in-domain set of 24K words.

4.2 Data Selection and Pruning

We built models selecting sentences below different thresholds on the WER, in-domain cross-entropy, or cross-entropy difference. For comparison, we also pruned our models using conventional count-cutoff and entropy pruning (Stolcke, 1998). During entropy pruning, we used a Good-Turing estimated model for computing the history marginals as the lower-order Kneser-Ney distributions are unsuitable for this purpose (Chelba et al., 2010).

We calculated the perplexity of each model on three test sets. We also tallied the number of model parameters (all n-gram probabilities plus all backoff weights). On TURKDEV, cross-entropy difference selection performed the best for all models sizes and for all training sets (figure 6). We also found cross-



(a) TWITTER (b) BLOG (c) USENET

Figure 7: Perplexity on TURKDEV varying the cross-entropy difference threshold.

entropy difference was the best on COMM, reducing perplexity by 10–20% relative compared to cross-entropy selection. Results on SPECIALISTS showed that WER and both forms of cross-entropy selection performed similarly. All three data selection methods were superior to count-cutoff or entropy pruning. We use cross-entropy difference selection for the remainder of this paper.

4.3 Model Order and Optimal Thresholds

We created 2-gram, 3-gram and 4-gram models on TWITTER, BLOG, and USENET using a range of cross-entropy difference thresholds. 4-gram models slightly outperformed 3-gram models (figure 7). The optimal threshold for 4-gram models were as follows: TWITTER 0.0, BLOG -0.4, and USENET -0.7. These thresholds resulted in using 20% of TWITTER, 5% of BLOG, and 1% of USENET.

4.4 Mixture Model

We created a mixture model using linear interpolation from the TWITTER, USENET and BLOG 4-gram models created with each set's optimal threshold. The mixture weights were optimized with respect to TURKDEV using SRILM. The final mixture weights were: TWITTER 0.42, BLOG 0.29, and USENET 0.29. Our final 4-gram mixture model had 43M total parameters and a compressed disk size of 316 MB.

5 Evaluation

In this section, we compare our mixture model against baseline models. We show performance with respect to usage in a typical AAC text entry interface based on word prediction.

5.1 Predictive Text Entry

Many AAC communication devices use word predictions. In a word prediction interface users type letters and the interface offers word completions based on the prefix of the current word and often the prior text. By selecting one of the predictions, the user can potentially save keystrokes as compared to typing out every letter of each word.

We assume a hypothetical predictive keyboard interface that displays five word predictions. Our keyboard makes predictions based on up to three words of prior context. Our keyboard predicts words even before the first letter of a new word is typed. As a user types letters, predictions are limited to words consistent with the typed letters. If the system makes a correct prediction, we assume it takes only one keystroke to enter the word and any following space.

We only predict words in our 63K word vocabulary (empty prediction slots are possible). We display a word even if it was already a proposed prediction for a shorter prefix of the current word. The first word in a sentence is conditioned on the sentence-start pseudo-word. If an out-of-vocabulary word is typed, the word is replaced in the language model's context with the unknown pseudo-word.

We evaluate our predictive keyboard using the common metric of keystroke savings (KS):

$$KS = \left(1 - \left(\frac{k_p}{k_a}\right)\right) \times 100\%,$$

where k_p is the number of keystrokes required with word predictions and k_a is the number of keystrokes required without word prediction.

5.2 Predictive Performance Experiment

We compared our mixture model using cross-entropy difference selection with three baseline models trained on all of TWITTER, SWITCHBOARD and TURKTRAIN. The baseline models were unpruned 4-gram models trained using interpolated modified Kneser-Ney smoothing. They had 72M, 5M, and 129K parameters respectively.

As shown in table 4, our mixture model performed the best on the three most AAC-like test sets (COMM, SPECIALISTS, and TURKTEST). The

LM	Test set	PPL	KS
Mixture	**COMM**	**47.9**	**62.5%**
Twitter	COMM	55.9	60.9%
Switchboard	COMM	151.1	54.4%
Turk	COMM	165.9	52.7%
Mixture	**SPECIALISTS**	**25.7**	**63.1%**
Twitter	SPECIALISTS	27.3	61.9%
Switchboard	SPECIALISTS	64.5	57.7%
Turk	SPECIALISTS	85.9	52.8%
Mixture	**TURKTEST**	**31.2**	**62.0%**
Twitter	TURKTEST	42.3	59.3%
Switchboard	TURKTEST	172.5	50.6%
Turk	TURKTEST	51.0	57.6%
Mixture	SWITCHTEST	174.3	52.8%
Twitter	SWITCHTEST	142.6	54.9%
Switchboard	**SWITCHTEST**	**79.2**	**58.8%**
Turk	SWITCHTEST	642.5	42.9%

Table 4: Perplexity (PPL) and keystroke savings (KS) of different language models on four test sets. The bold line shows the best performing language model on each test set.

mixture model provided substantial increases in keystroke savings compared to a model trained solely on Switchboard. The mixture model also performed better than simply training a model on a large amount of Twitter data. The model trained on only 24K words of Turk data did surprisingly well given its extremely limited training data.

Our Switchboard model performed the best on SWITCHTEST with a keystroke savings of 58.8%. For comparison, past work reported a keystroke savings of 55.7% on SWITCHTEST using a 3-gram model trained on Switchboard (Trnka et al., 2009). While our mixture model performed less well on SWITCHTEST (52.8%), it is likely the other three test sets better represent AAC communications.

5.3 Larger Mixture Model Experiment

Our mixture language model used the best thresholds with respect to TURKDEV. This resulted in throwing away most of the training data. This might be suboptimal in practice if an AAC user's communications are somewhat different or more diverse than the language generated by the Turk workers.

We trained a series of mixture models in which we varied the cross-entropy difference thresholds

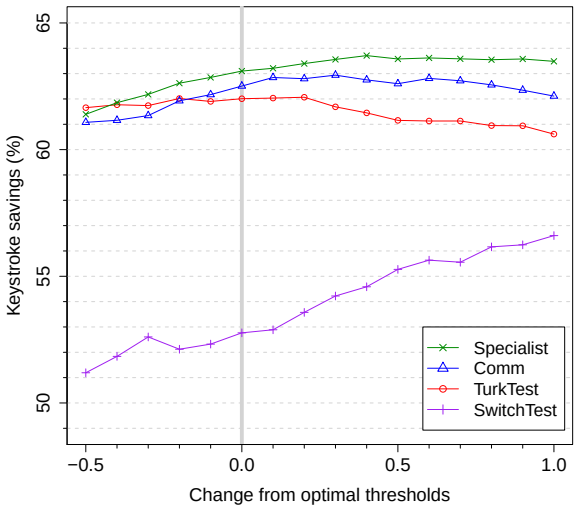


Figure 8: Keystroke savings on mixture models varying a constant added to the optimal thresholds with respect to TURKDEV.

by adding a constant to all three thresholds. The mixture weights for each new model were optimized with respect to TURKDEV. Using somewhat larger models did improve keystroke savings for all test sets except for TURKTEST (figure 8). However, using too large thresholds eventually hurt performance except on SWITCHTEST. Performance on SWITCHTEST steadily increased from 52.8% to 56.6%. These gains however came at the cost of bigger models. The model using +1.0 of the optimal thresholds had 384M parameters and a compressed size of 3.0 GB.

6 Discussion

Given the ethical implications of collecting messages from actual AAC users, it is unlikely that a large corpus of genuine AAC messages will ever be available to researchers. An important finding in this paper is that crowdsourcing can be an effective way to obtain surrogate data for improving AAC language models. Another finding is that Twitter provides a continuous stream of large amounts of very AAC-like data. Twitter also has the advantage of allowing models to be continually updated to reflect current events, new vocabulary, etc.

6.1 Limitations and Implications

We collected data from a large number of workers, some of whom may have written only a single com-

munication. This may have resulted in more messages about simple situations and perceived needs which could differ from true AAC usage.

Our data does not contain long-term two-sided conversations. Thus it may not be as useful for evaluating techniques that adapt to past messages or that use the conversation partner's communications.

We asked workers to imagine they were using a scanning-style AAC device. We believe this led workers to presume they would require assistance in many routine physical tasks. Our workers were (presumably) without cognitive or language impairments. Thus our collection is more representative of one subgroup of AAC communicators (scanning users with normal cognitive function and language skills). By modifying the situation given to workers, it is likely we can expand our collection to better represent other groups of AAC users, such as those using predictive keyboards or eye-trackers. However, obtaining data representative of users with cognitive or language impairments via crowdsourcing would probably be difficult.

While we were unable to obtain real AAC messages for testing, we believe the COMM and SPECIALIST test sets provide a good indication of the real-world potential for our methods. Our collected Turk data was compared with reported data from actual AAC users (though this comparison was necessarily coarse-grained). We hope that by releasing our data and models it may be possible for those privy to real AAC communications to validate and report about the techniques described in this paper.

We evaluated our models in terms of perplexity and keystrokes savings within the auspices of a predictive keyboard. Further work is needed to investigate how our numeric gains translate to real-world benefits to users. However, past work indicates more accurate predictions do in fact yield improvements in human performance (Trnka et al., 2009).

Finally, while the predictive keyboard is a commonly studied interface, it is not appropriate for all AAC users. Eye-tracker users may prefer an interface such as Dasher (Ward and MacKay, 2002). Single-switch users may prefer an interface such as Nomon (Broderick and MacKay, 2009). Any AAC interface based on word- or letter-based predictions stands to benefit from the methods described in this paper.

7 Conclusions

In this paper we have shown how workers' creativity on a microtask crowdsourcing market can be used to create fictional but plausible AAC communications. We have demonstrated that these messages model conversational AAC better than the currently used datasets based on telephone conversations or newswire text. We used our new crowdsourced dataset to intelligently select sentences from Twitter, blog and Usenet data.

We compared a variety of different techniques for intelligent training data selection. We found that even for our small amount of in-domain data, the recently proposed cross-entropy difference method was consistently the best (Moore and Lewis, 2010). Finally, compared to a model trained only on Switchboard, our best performing model reduced perplexity by 60-82% relative on three AAC-like test sets. This translated to a potential keystroke savings in a predictive keyboard interface of 5–11%.

In conclusion, we have shown how to create long-span AAC language models using openly available resources. Our models significantly outperform models trained on the commonly used data sources of telephone transcripts and newswire text. To aid other researchers, we have publicly released our crowdsourced AAC collection, word lists and best-performing models. We hope complementary techniques such as topic modeling and language model adaptation will provide additive gains to those obtained by training models on large amounts of AAC-like data. We plan to use our models to design and test new interfaces that enable faster communication for AAC users.

Acknowledgments

We thank Keith Trnka and Horabail Venkatagiri for their assistance. This work was supported by the Engineering and Physical Sciences Research Council (grant number EP/H027408/1).

References

Bruce Baker, Katya Hill, and Richard Devylder. 2000. Core vocabulary is the same across environments. In *California State University at Northridge Conference*.

David R. Beukelman, Kathryn M. Yorkston, Miguel Poblete, and Carlos Naranjo. 1984. Frequency of

word occurrence in communication samples produced by adult communication aid users. *Journal of Speech and Hearing Disorders*, 49:360–367.

Tamara Broderick and David J. C. MacKay. 2009. Fast and flexible selection with a single switch. *PLoS ONE*, 4(10):e7481.

Ivan Bulyko, Mari Ostendorf, Manhung Siu, Tim Ng, Andreas Stolcke, and Özgür Çetin. 2007. Web resources for language modeling in conversational speech recognition. *ACM Transactions on Speech and Language Processing*, 5(1):1–25.

Kevin Burton, Akshay Java, and Ian Soboroff. 2009. The ICWSM 2009 Spinn3r dataset. In *Proceedings of the 3rd Annual Conference on Weblogs and Social Media*.

Ciprian Chelba, Thorsten Brants, Will Neveitt, and Peng Xu. 2010. Study on interaction between entropy pruning and Kneser-Ney smoothing. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2422–2425.

Stanley F. Chen and Joshua T. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University.

Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. 2002. Toward a unified approach to statistical language modeling for chinese. *ACM Transactions on Asian Language Information Processing*, 1:3–33.

Nestor Garay-Vitoria and Julio Abascal. 2006. Text prediction systems: A survey. *Universal Access in the Information Society*, 4:188–203.

Nestor Garay-Vitoria and Julio González-Abascal. 1997. Intelligent word-prediction to enhance text input rate. In *Proceedings of the 2nd ACM International Conference on Intelligent User Interfaces*, pages 241–244.

J.J. Godfrey, E.C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 517–520.

David Graff, Roni Rosenfeld, and Doug Pau. 1995. CSR-III text. Linguistic Data Consortium, Philadelphia, PA, USA.

David Graff. 2003. English gigaword corpus. Linguistic Data Consortium, Philadelphia, PA, USA.

Sheri Hunnicutt. 1989. Using syntactic and semantic information in a word prediction aid. In *Proceedings of the 1st European Conference on Speech Communication and Technology*, pages 1191–1193.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

Gregory W. Lesher and Gerard J. Rinkus. 2002. Domain-specific word prediction for augmentative communication. In *Proceedings of the RESNA 2002 Annual Conference*.

Jianhua Li and Graeme Hirst. 2005. Semantic knowledge in word completion. In *Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 121–128.

Sung-Chien Lin, Chi-Lung Tsai, Lee-Feng Chien, Ker-Jiann Chen, and Lin-Shan Lee. 1997. Chinese language model adaptation based on document classification and multiple domain-specific language models. In *Proceedings of the 5th European Conference on Speech Communication and Technology*, pages 1463–1466.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association of Computational Linguistics*, pages 220–224.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Proceedings of HLT-NAACL 2010*, pages 172–180.

Cyrus Shaoul and Chris Westbury. 2009. A USENET corpus (2005-2009). University of Alberta, Canada.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the 7th Annual International Conference on Spoken Language Processing*, pages 901–904.

Keith Trnka, Debra Yarrington, and Christopher Pennington. 2006. Topic modeling in fringe word prediction for AAC. In *Proceedings of the 11th ACM International Conference on Intelligent User Interfaces*, pages 276–278.

Keith Trnka, John McCaw, Debra Yarrington, Kathleen F. McCoy, and Christopher Pennington. 2009. User interaction with word prediction: The effects of prediction quality. *ACM Transactions on Accessible Computing*, 1:17:1–17:34.

Keith Trnka. 2008. Adaptive language modeling for word prediction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Student Research Workshop*, pages 61–66.

Horabail Venkatagiri. 1999. Efficient keyboard layouts for sequential access in augmentative and alternative communication. *Augmentative and Alternative Communication*, 15(2):126–134.

Tonio Wandmacher and Jean-Yves Antoine. 2007. Methods to integrate a language model with semantic

information for a word prediction component. *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 506–513.

Tonio Wandmacher, Jean-Yves Antoine, Franck Poirier, and Jean-Paul Départe. 2008. SIBYLLE, an assistive communication system adapting to the context and its user. *ACM Transactions on Accessible Computing*, 1:6:1–6:30.

D. J. Ward and D. J. C. MacKay. 2002. Fast hands-free writing by gaze direction. *Nature*, 418(6900):838.

Using Syntactic and Semantic Structural Kernels for Classifying Definition Questions in Jeopardy!

Alessandro Moschitti[†] **Jennifer Chu-Carroll**[‡] **Siddharth Patwardhan**[‡]

James Fan[‡] **Giuseppe Riccardi**[†]

[†]Department of Information Engineering and Computer Science
University of Trento, 38123 Povo (TN), Italy
{moschitti,riccardi}@disi.unitn.it
[‡]IBM T.J. Watson Research Center P.O. Box 704, Yorktown Heights, NY 10598, U.S.A.
{jencc,siddharth,fanj}@us.ibm.com

Abstract

The last decade has seen many interesting applications of Question Answering (QA) technology. The Jeopardy! quiz show is certainly one of the most fascinating, from the viewpoints of both its broad domain and the complexity of its language. In this paper, we study kernel methods applied to syntactic/semantic structures for accurate classification of Jeopardy! definition questions. Our extensive empirical analysis shows that our classification models largely improve on classifiers based on word-language models. Such classifiers are also used in the state-of-the-art QA pipeline constituting Watson, the IBM Jeopardy! system. Our experiments measuring their impact on Watson show enhancements in QA accuracy and a consequent increase in the amount of money earned in game-based evaluation.

1 Introduction

Question Answering (QA) is an important research area of Information Retrieval applications, which requires the use of core NLP capabilities, such as syntactic and semantic processing for a more effective user experience. While the development of most existing QA systems are driven by organized evaluation efforts such as TREC (Voorhees and Dang, 2006), CLEF (Giampiccolo et al., 2007), and NTCIR (Sasaki et al., 2007), there exist efforts that leverage data from popular quiz shows, such as Who Wants to be a Millionaire (Clarke et al., 2001; Lam et al., 2003) and Jeopardy! (Ferrucci et al., 2010), to demonstrate the generality of the technology.

Jeopardy! is a popular quiz show in the US which has been on the air for 27 years. In each game, three contestants compete for the opportunity to answer 60 questions in 12 categories of 5 questions each. Jeopardy! questions cover an incredibly broad domain, from science, literature, history, to popular culture. We are drawn to Jeopardy! as a test bed for open-domain QA technology due to its broad domain, complex language, as well as the emphasis on accuracy, confidence, and speed during game play.

While the vast majority of Jeopardy! questions are factoid questions, we find several other types of questions in the Jeopardy! data, which can benefit from specialized processing in the QA system. The additional processing in these questions complements that of the factoid questions to achieve improved overall QA performance. Among the various types of questions handled by the system are *definition questions* shown in the examples below:

(1) GON TOMORROW*: It can be the basket below a hot-air balloon or a flat-bottomed boat used on a canal* (answer: gondola);

(2) I LOVE YOU, "MIN"*: Overbearing* (answer: domineering);

(3) INVEST*: From the Latin for "year", it's an investment or retirement fund that pays out yearly* (answer: an annuity)

where the upper case text indicates the Jeopardy! category for each question[1].

Several characteristics of this class of questions warrant special processing: first, the clue (question)

[1]A Jeopardy! category indicates a theme is common among its 5 questions.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 712–724,
Edinburgh, Scotland, UK, July 27–31, 2011.

often aligns well with dictionary entries, making dictionary resources potentially effective. Second, these clues often do not indicate an answer type, which is an important feature for identifying correct answers in factoid questions (in the examples above, only (3) provided an answer type, "fund"). Third, definition questions are typically shorter in length than the average factoid question. These differences, namely the shorter clue length and the lack of answer types, make the use of a specialized machine learning model potentially promising for improving the overall system accuracy. The first step for handling definitions is, of course, the automatic separation of definitions from other question types, which is not a simple task in the Jeopardy! domain. For instance, consider the following example which is a variation of (3) above:

(4) INVEST*: From the Latin for "year", an annuity is an investment or retirement fund that pays out this often* (answer: yearly)

Even though the clue is nearly identical to (3), the clue does *not* provide a definition for the answer *yearly*, although at first glance we may have been misled. The source of complexity is given by the fact that Jeopardy! clues are not phrased in interrogative form as questions typically are. This complicates the design of definition classifiers since we cannot directly use either typical structural patterns that characterize definition/description questions, or previous approaches, e.g. (Ahn et al., 2004; Kaisser and Webber, 2007; Blunsom et al., 2006). Given the complexity and the novelty of the task, we found it useful to exploit the kernel methods technology. This has shown state-of-the-art performance in Question Classification (QC), e.g. (Zhang and Lee, 2003; Suzuki et al., 2003; Moschitti et al., 2007) and it is very well suited for engineering feature representations for novel tasks.

In this paper, we apply SVMs and kernel methods to syntactic/semantic structures for modeling accurate classification of Jeopardy! definition questions. For this purpose, we use several levels of linguistic information: word and POS tag sequences, dependency, constituency and predicate argument structures and we combined them using state-of-the-art structural kernels, e.g. (Collins and Duffy, 2002; Shawe-Taylor and Cristianini, 2004; Moschitti, 2006). The extensive empirical analysis of several advanced models shows that our best model, which combines different kernels, improves the F1 of our baseline model by 67% relative, from 40.37 to 67.48. Surprisingly, with respect to previous findings on standard QC, e.g. (Zhang and Lee, 2003; Moschitti, 2006), the Syntactic Tree Kernel (Collins and Duffy, 2002) is not effective whereas the exploitation of partial tree patterns proves to be essential. This is due to the different nature of Jeopardy! questions, which are not expressed in the usual interrogative form.

To demonstrate the benefit of our question classifier, we integrated it into our Watson by coupling it with search and candidate generation against specialized dictionary resources. We show that in end-to-end evaluations, Watson with kernel-based definition classification and specialized definition question processing achieves statistically significant improvement compared to our baseline systems.

In the reminder of this paper, Section 2 describes Watson by focusing on the problem of definition question classification, Section 3 describes our models for such classifiers, Section 4 presents our experiments on QC, whereas Section 5 shows the final impact on Watson. Finally, Section 6 discusses related work and Section 7 derives the conclusions.

2 Watson: The IBM Jeopardy! System

This section gives a quick overview of Watson and the problem of classification of definition questions, which is the focus of this paper.

2.1 Overview

Watson is a massively parallel probabilistic evidence-based architecture for QA (Ferrucci et al., 2010). It consists of several major stages for underlying sub-tasks, including analysis of the question, retrieval of relevant content, scoring and ranking of candidate answers, as depicted in Figure 1. In the rest of this section, we provide an overview of Watson, focusing on the task of answering definitional questions.

Question Analysis: The first stage of the pipeline, it applies several analytic components to identify key characteristics of the question (such as answer

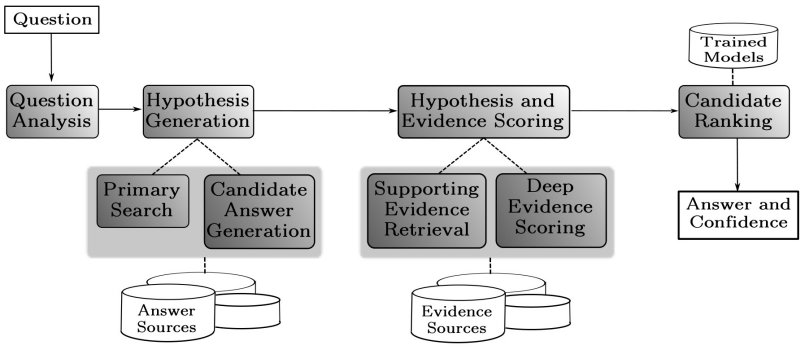


Figure 1: Overview of Watson

type, question classes, etc.) used by later stages of the Watson pipeline. Various general purpose NLP components, such as a parser and named entity detector, are combined with task-specific modules for this analysis.

The task-specific analytics include several QC components, which determine if the question belongs to one or more broad "question classes". These question classes can influence later stages of the Watson pipeline. For instance, a question detected as an *abbreviation* question can invoke specialized candidate generators to produce possible expansions of the abbreviated term in the clue. Similarly, the question classes can impact the methods for answer scoring and the machine learning models used for ranking candidate answers. The focus of this paper is on the *definition* class, which is described in the next section.

Hypothesis Generation: Following question analysis, the Watson pipeline searches its document collection for relevant documents and passages that are likely to contain the correct answer to the question. This stage of the pipeline generates search queries based on question analysis results, and obtains a ranked list of documents and passages most relevant to the search queries. A variety of candidate generation techniques are then applied to the retrieved results to produce a set of candidate answers.

Information obtained from question analysis can be used to influence the search and candidate generation processes. The question classes detected during question analysis can focus the search towards specific subsets of the corpus. Similarly, during candidate generation, strategies used to generate the set of candidate answers are selected based on the detected question classes.

Hypothesis and Evidence Scoring: A wide variety of answer scorers are then used to gather evidence supporting each candidate answer as the correct answer to the given question. The scorers include both context dependent as well as context independent scorers, relying on various structured and unstructured resources for their supporting evidence.

Candidate Ranking: Finally, machine learning models are used to weigh the gathered evidence and rank the candidate answers. The models generate a ranked list of answers each with an associated confidence. The system can also choose to refrain from answering a question if it has low confidence in all candidates. This stage of the pipeline employs several machine learning models specially trained to handle various types of questions. These models are trained using selected feature sets based on question classes and candidate answers are "routed" to the appropriate model according to the question classes detected during question analysis.

2.2 Answering Definition Questions

Among the many question classes that Watson identifies and leverages for special processing, of particular interest for this paper is the class we refer to as *definition* questions. These are questions whose clue texts contain one or more definitions of the correct answer. For instance, in example (3), the main clause in the question corresponds to a dictionary definition of the correct answer (*annuity*). Looking up this definition in dictionary resources could enable us to answer this question correctly and with high confidence. This suggests that special process-

ing of such definition questions could allow us to hone in on the correct answer through processes different from those used for other types of questions.

This paper explores strategies for definition question processing to improve overall question answering performance. A key challenge we have to address is that of accurate recognition of such questions. Given an input question the Watson question analysis stage uses a definition question recognizer to detect this specific class of questions. We explore several approaches for recognition, including a rule based approach and a variety of statistical models.

Questions that are recognized as definition questions invoke search processes targeted towards dictionary-like sources in our system. We use a variety of such sources, such as standard English dictionaries, Wiktionary, WordNet, etc. After gathering supporting evidence for candidate answers extracted from these sources, our system routes the candidates to definition-specific candidate ranking models, which have been trained with selected feature sets.

The following sections present a description and evaluation of our approach for identifying and answering definition questions.

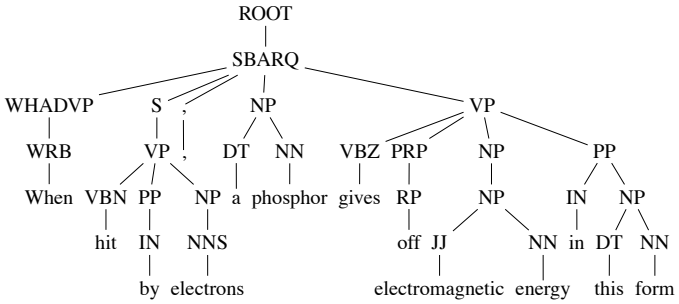


Figure 2: Constituency Tree

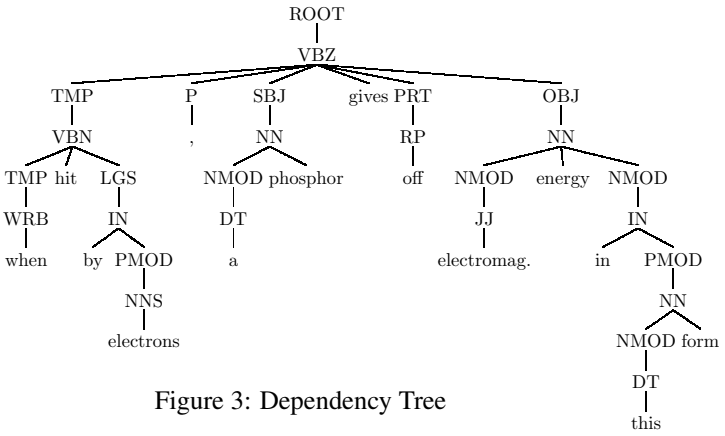


Figure 3: Dependency Tree

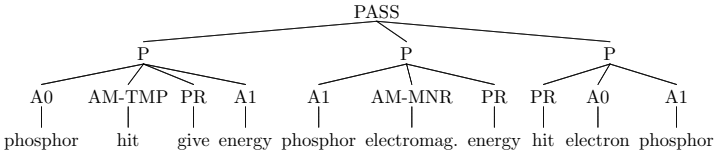


Figure 4: A tree encoding a Predicate Argument Structure Set

3 Kernel Models for Question Classification

Previous work (Zhang and Lee, 2003; Suzuki et al., 2003; Blunsom et al., 2006; Moschitti et al., 2007) as shown that syntactic structures are essential for QC. Given the novelty of both the domain and the type of our classification items, we rely on kernel methods to study and design effective representations. Indeed, these are excellent tools for automatic feature engineering, especially for unknown tasks and domains. Our approach consists of using SVMs and kernels for structured data applied to several types of structural lexical, syntactic and shallow semantic information.

3.1 Tree and Sequence Kernels

Kernel functions are implicit scalar products between data examples (i.e. questions in our case) in the very high dimensional space of substructures, where each of the latter is a component of the implicit vectors associated with the examples.

Although several kernels for structured data have been developed (see Section 6), the main distinctions in terms of feature spaces is given by the following three different kernels:

• Sequence Kernels (SK); we implemented the discontinuous string kernels described in (Shawe-Taylor and Cristianini, 2004). This allows for representing a string of symbols in terms of its possible substrings with gaps, i.e. an arbitrary number of symbols can be skipped during the generation of a substring. The symbols we used in the sequential descriptions of questions are words and part-of-speech tags (in two separate sequences). Consequently, all possible multiwords with gaps are features of the implicitly generated vector space.

• Syntactic Tree Kernel (STK) (Collins and Duffy, 2002) applied to constituency parse trees. This generates all possible tree fragments as features with the conditions that sibling nodes from the original trees cannot be separated. In other words, substructures are composed by atomic building blocks corresponding to nodes along with all their direct children. These, in case of a syntactic parse tree, are complete production rules of the associated parser grammar[2].

• Partial Tree Kernel (PTK) (Moschitti, 2006) applied to both constituency and dependency parse trees. This generates all possible tree fragments, as above, but sibling nodes can be separated (so they can be part of different tree fragments). In other words, a fragment is any possible tree path, from whose nodes other tree paths can depart. Consequently, an extremely rich feature space is generated. Of course, PTK subsumes STK but sometimes the latter provides more effective solutions as the number of irrelevant features is smaller as well.

When applied to sequences and tree structures, the kernels discussed above produce many different kinds of features. Therefore, the design of appropriate syntactic/semantic structures determines the representational power of the kernels. Hereafter, we show the models we used.

3.2 Syntactic Semantic Structures

We applied the above kernels to different structures. These can be divided in sequences of words (WS) and part of speech tags (PS) and different kinds of trees. For example, given the non-definition Jeopardy! question:

(5) GENERAL SCIENCE*: When hit by electrons, a phosphor gives off electromagnetic energy in this form.* (answer: light or photons),

we use the following sequences:

WS: [when][hit][by][electrons][,][a][phosphor][gives][off][electromagnetic][energy][in][this][form]

PS: [wrb][vbn][in][nns][,][dt][nn][vbz][rp][jj][nn][in][dt][nn]

Additionally, we use constituency trees (CTs), see Figure 2 and dependency structures converted into the dependency trees (DTs), e.g. shown in Figure 3. Note that, the POS-tags are central nodes, the grammatical relation label is added as a father node and all the relations with the other nodes are described by means of the connecting edges. Words are considered additional children of the POS-tag nodes (in this case the connecting edge just serves to add a lexical feature to the target POS-tag node).

Finally, we also use predicate argument structures generated by verbal and nominal relations according to PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004). Given the target sentence, the set of its predicates are extracted and converted into a forest, then a fake root node, PAS, is used to connect these trees. For example, Figure 4 illustrates a Predicate Argument Structures Set (PASS) encoding two relations, *give* and *hit*, as well as the nominalization *energy* along with all their arguments.

4 Experiments on Definition Question Classification

In these experiments, we study the role of kernel technology for the design of accurate classification of definition questions. We build several classifiers based on SVMs and kernel methods. Each classifier uses advanced syntactic/semantic structural features and their combination. We carry out an extensive comparison in terms of F1 between the different models on the Jeopardy! datasets.

4.1 Experimental Setup

Corpus: the data for our QC experiments consists of a randomly selected set of 33 Jeopardy! games[3]. These questions were manually annotated based on whether or not they are considered definitional. This resulted in 306 definition and 4964 non-definition clues. Each test set is stored in a separate file consisting of one line per question, which contains tab-separated clue information and the Jeopardy! category, e.g. INVEST in example (4).

Tools: for SVM learning, we used the SVMLight-TK software[4], which includes structural kernels in SVMLight (Joachims, 1999)[5]. For generating con-

[2]From here the name syntactic tree kernels

[3]Past Jeopardy! games can be downloaded from http://www.j-archive.com.

[4]Available at `http://dit.unitn.it/~moschitt`

[5]`http://svmlight.joachims.org`

stituency trees, we used the Charniak parser (Charniak, 2000). We also used the syntactic–semantic parser by Johansson and Nugues (2008) to generate dependency trees (Mel'čuk, 1988) and predicate argument trees according to the PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) frameworks.

Baseline Model: the first model that we used as a baseline is a rule-based classifier (RBC). The RBC leverages a set of rules that matches against lexical and syntactic information in the clue to make a binary decision on whether or not the clue is considered definitional. The rule set was manually developed by a human expert, and consists of rules that attempt to identify roughly 70 different constructs in the clues. For instance, one of the rules matches the parse tree structure for "It's X or Y", which will identify example (1) as a definition question.

Kernel Models: we apply the kernels described in Section 3 to the structures extracted from Jeopardy! clues. In particular, we design the following models: BOW, i.e. linear kernel on bag-of-words from the clues; WSK, PSK and CSK, i.e. SK applied to the word and POS-tag sequences from the clues, and the word sequence taken from the question categories, respectively; STK-CT, i.e. STK applied to CTs of the clue; PTK-CT and PTK-DT, i.e. PTK applied to CTs and DTs of the clues, respectively; PASS, i.e. PTK applied to the Predicate Argument Structure Set extracted from the clues; and RBC, i.e. a linear kernel applied to the vector only constituted by the 1/0 output of RBC.

Learning Setting: there is no particular parameterization. Since there is an imbalance between positive and negative examples, we used a Precision/Recall trade-off parameter in SVM-Light-TK equal to 5.[6]

Measures: the performance is measured with Precision, Recall and F1-measure. We estimated them by means of Leave-One-Out[7] (LOO) on the question set.

4.2 Results and Discussion

Table 1 shows the performance obtained using different kernels (feature spaces) with SVMs. We note that: first, RBC has good Recall but poor Precision. This is interesting since, on one hand, these results validate the complexity of the task: in order to capture the large variability of the positive examples, the rules developed by a skilled human designer are unable to be sufficiently precise to limit the recognition to those examples. On the other hand, RBC, being a rather different approach from SVMs, can be successfully exploited in a joint model with them.

Kernel Space	Prec.	Rec.	F1
RBC	28.27	70.59	**40.38**
BOW	47.67	46.73	47.20
WSK	47.11	50.65	48.82
STK-CT	50.51	32.35	39.44
PTK-CT	47.84	57.84	**52.37**
PTK-DT	44.81	57.84	50.50
PASS	33.50	21.90	26.49
PSK	39.88	45.10	42.33
CSK	39.07	77.12	**51.86**

Table 1: Kernel performance using leave-one-out cross-validation.

Second, BOW yields better F1 than RBC but it does not generalize well since its F1 is still low. When n-grams are also added to the model by means of WSK, the F1 improves by about 1.5 absolute points. As already shown in (Zhang and Lee, 2003; Moschitti et al., 2007), syntactic structures are needed to improve generalization.

Third, surprisingly with respect to previous work, STK applied to CT[8] provides accuracy lower than BOW, about 8 absolute points. The reason is due to the different nature of the Jeopardy! questions: large syntactic variability reduces the probability of finding general and well formed patterns, i.e. structures generated by entire production rules. This suggests that PTK, which can capture patterns derived from partial production rules, can be more effective. Indeed, PTK-CT achieves the highest F1, outperforming WSK also when used with a different syntactic paradigm, i.e. PTK-DT.

Next, PSK and PASS provide a lower accuracy but they may be useful in kernel combinations as they can complement the information captured by the other models. Interestingly, CSK alone is rather effective for classifying definition questions. We be-

[6] We have selected 5 as a reasonable value, which kept balanced Precision and Recall on a validation set.

[7] LOO applied to a corpus of N instances consists in training on $N-1$ examples and testing on the single held-out example. This process is repeated for all instances.

[8] Applying it to DT does not make much sense as already pointed out in (Moschitti, 2006).

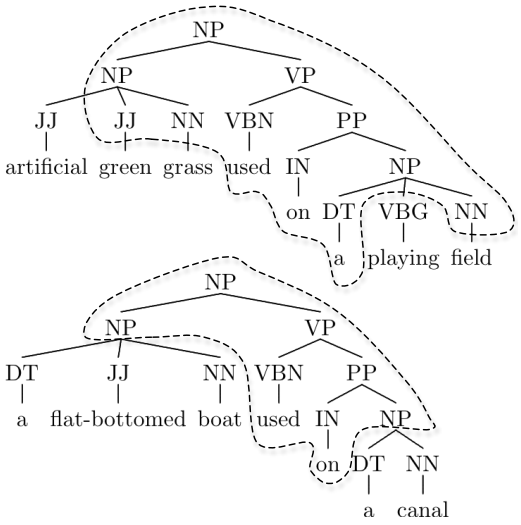


Figure 5: Similarity according to PTK and STK

lieve this is because definition questions are sometimes clustered into categories such as 4-LETTER WORDS or BEGINS WITH "B".

Moreover, we carried out qualitative error analysis on the PTK and STK outcome, which supported our initial hypothesis. Let us consider the bottom tree in Figure 5 in the training set. The top tree is a test example correctly classified by PTK but incorrectly classified by STK. The dashed line in the top tree contains the largest subtree matched by PTK (against the bottom tree), whereas the dashed line in the bottom tree indicates the largest subtree matched by STK (against the top tree). As the figure shows, PTK can exploit a larger number of partial patterns.

Finally, the above points suggest that different kernels produce complementary information. It is thus promising to experiment with their combinations. The joint models can be simply built by summing kernel functions together. The results are shown in Table 2. We note that: (i) CSK complements the WSK information, achieving a substantially better result, i.e. 62.95; (ii) PTK-CT+CSK performs even better than WSK+CSK (as PTK outperforms WSK); and (iii) adding RBC improves further on the above combinations, i.e. 68.11 and 67.32, respectively. This evidently demonstrates that RBC captures complementary information. Finally, more complex kernels, especially the overall kernel summation, do not seem to improve the performance. This is also confirmed by the PASS results derived in (Moschitti et al., 2007) on TREC QC.

Kernel Space	Prec.	Rec.	F1
WSK+CSK	70.00	57.19	62.95
PTK-CT+CSK	69.43	60.13	64.45
PTK-CT+WSK+CSK	68.59	62.09	**65.18**
CSK+RBC	47.80	74.51	58.23
PTK-CT+CSK+RBC	59.33	74.84	65.79
BOW+CSK+RBC	60.65	73.53	66.47
PTK-CT+WSK+CSK+RBC	67.66	66.99	**67.32**
PTK-CT+PASS+CSK+RBC	62.46	71.24	66.56
WSK+CSK+RBC	69.26	66.99	**68.11**
ALL	61.42	67.65	64.38

Table 2: Performance of Kernel Combinations using leave-one-out cross-validation.

5 Experiments on the Jeopardy System

Since the kernel-based classifiers perform substantially better than RBC, we incorporate the PTK-CT+WSK+CSK model[9] into Watson for definition classification and evaluated the QA performance against two baseline systems. For the end-to-end experiments, we used Watson's English Slot Grammar parser (McCord, 1980) to generate the constituency trees. The component level evaluation shows that we achieved comparable performance as previously discussed with ESG.

5.1 Experimental Setup

We integrated the classifier into the question analysis module, and incorporated additional components to search against dictionary resources and extract candidate answers from these search results when a question is classified as definitional. In the final machine learning models, a separate model is trained for definition questions to enable scoring tailored to the specific characteristics of those questions.

Based on our manually annotated gold standard, less than 10% of Jeopardy! questions are classified as definition questions. Due to their relatively low frequency we conduct two types of evaluations. The first is *definition-only evaluation*, in which we apply our definition question classifier to identify a large

[9]Since we aim to compare a purely statistical approach to the rule-based approach, we did not experiment with the model that uses RBC as a feature in our end-to-end experiments.

set of definition questions and evaluate the end-to-end system's performance on this large set of questions. These results enable us to draw statistically significant conclusions about our approach to addressing definition questions.

The second type of evaluation is *game-based evaluation*, which assesses the impact of our definition question processing on Watson performance while preserving the natural distribution of these question types in Jeopardy! data. Game-based evaluations situate the system's performance on definition questions relative to other types of questions, and enable us to gauge the component's contributions in a game-based setting.

For both evaluation settings, three configurations of Watson are used as follows:

- the **NoDef** system, in which Watson is configured without definition classification and processing, thereby treating all definition questions as regular factoid questions;
- the **StatDef** system, which leverages the statistical classifier and subsequent definition specific search and candidate generation components as described above; and
- the **RuleDef** system, in which Watson adopts RBC and employs the same additional definition search and candidate generation components as the StatDef system.

For the definition-only evaluation, we selected all questions recognized as definitional by the statistical classifier from roughly 1000 unseen games (60000 questions), resulting in a test set of 1606 questions. Due to the size of the initial set, it is impractical to manually create a gold standard for measuring Precision and Recall of the classifier. Instead, we compare the StatDef system against the NoDef on these 1606 questions using two metrics: accuracy, defined as the percentage of questions correctly answered, and p@70, the system's Precision when answering only the top 70% most confident questions. P@70 is an important metric in Jeopardy! game play as well as in real world applications where the system may refrain from answering a question when it is not confident about any of its answers. Since RBC identifies significantly more definition questions, we started with an initial set of roughly 300 games, from which the RBC identified 1875 questions as definitional. We compared the RuleDef system's performance on these questions against the NoDef baseline using the accuracy and p@70 metrics.

	NoDef	**StatDef**	**NoDef**	**RuleDef**
# Questions	1606	1606	1875	1875
Accuracy	63.76%	65.57%	56.64%	57.51%
P@70	82.22%	84.53%	72.73%	74.87%

Table 3: Definition-Only Evaluation Results

For the game-based evaluation, we randomly selected 66 unseen Jeopardy! games, consisting of 3546 questions after excluding audio/visual questions.[10] We contrast the StatDef system performance against that of NoDef and RuleDef along several dimensions: accuracy and p@70, described above, as well as earnings, the average amount of money earned for each game.

5.2 Definition-Only Evaluation

For the definition-only evaluation, we compared the StatDef system against the NoDef system on a set of 1606 questions that the StatDef system classified as definitional. The results are shown in the first two columns in Table 3. To contrast the gain obtained by the StatDef system against that achieved by the RuleDef system, we ran the RuleDef system over the 1875 questions identified as definitional by the rule-based classifier. We contrast the RuleDef system performance with that of the NoDef system, as shown in the last two columns in Table 3.

Our results show that based on both evaluation metrics, StatDef improved upon the NoDef baseline more than RuleDef improved on the same baseline system. Furthermore, for the accuracy metric where all samples are paired and independent, the difference in performance between the StatDef and NoDef systems is statistically significant at $p<0.05$, while that between the RuleDef and NoDef systems is not.

5.3 Game-Based Evaluation

The game-based evaluation was carried out on 66 unseen games (roughly 3500 questions). Of these

[10] Audio/visual questions are those accompanied by either an image or an audio clip. The text portions of these questions are often insufficient for identifying the correct answers.

	# Def Q's	Accuracy	P@70	Earnings
NoDef	0	69.71%	86.79%	$24,818
RuleDef	480	69.23%	86.31%	$24,397
StatDef	131	69.85%	87.19%	$25,109

Table 4: Game-Based Evaluation Results

questions, the StatDef system classified 131 of them as definitional while the RuleDef system identified 480 definition questions. Both systems were compared against the NoDef system using the accuracy, p@70, and earnings metric computed over all questions, as shown in Table 4.

Our results show that even though in the definition-only evaluation both the RuleDef and StatDef systems outperformed the NoDef baseline, in our game-based evaluation, the RuleDef system performed *worse* than the NoDef baseline. The lowered performance is due to the fact that the Precision of the RBC is much lower than that of the statistical classifier, and the special definition processing applied to questions that are erroneously classified as definitional was harmful. Our evaluation of this false positive set showed that its accuracy dropped by 6% compared to the NoDef system. On the other hand, the StatDef system outperformed the two other systems, and its accuracy improvement upon the RuleDef system is statistically significant at $p<0.05$.

6 Related Work

Our paper studies the use of advanced representation for QC in the Jeopardy! domain. As previously mentioned Jeopardy! questions are stated as affirmative sentences, which are different from the typical QA questions. For the design of our models, we have carefully taken into account previous work. This shows that semantics and syntax are essential to retrieve precise answers, e.g (Hickl et al., 2006; Voorhees, 2004; Small et al., 2004).

We focus on definition questions, which typically require more complex processing than factoid questions (Blair-Goldensohn et al., 2004; Chen et al., 2006; Shen and Lapata, 2007; Bilotti et al., 2007; Moschitti et al., 2007; Surdeanu et al., 2008; Echihabi and Marcu, 2003). For example, language models were applied to definitional QA in (Cui et al., 2005) to learn soft pattern models based on bigrams. Other related work, such as (Sasaki, 2005; Suzuki et al., 2002), was also very tied to bag-of-words features. Predicate argument structures have been mainly used for reranking (Shen and Lapata, 2007; Bilotti et al., 2007; Moschitti et al., 2007; Surdeanu et al., 2008).

Our work and methods are similar to (Zhang and Lee, 2003; Moschitti et al., 2007), which achieved the state-of-the-art in QC by applying SVMs along with STK-CT. The results were derived by experimenting with a TREC dataset[11](Li and Roth, 2002), reaching an accuracy of 91.8%. However, such data refers to typical instances from QA, whose syntactic patterns can be easily generalized by STK. In contrast, we have shown that STK-CT is not effective for our domain, as it presents very innovative elements: questions in affirmative and highly variable format. Thus, we employed new methods such as PTK, dependency structures, multiple sequence kernels including category information and many combinations.

Regarding the use of Kernel Methods, there is a considerably large body of work in Natural Language Processing, e.g. regarding syntactic parsing (Collins and Duffy, 2002; Kudo et al., 2005; Shen et al., 2003; Kudo and Matsumoto, 2003; Titov and Henderson, 2006; Toutanova et al., 2004), named entity recognition and chunking (Cumby and Roth, 2003; Daumé III and Marcu, 2004), relation extraction (Zelenko et al., 2002; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Zhang et al., 2005; Bunescu, 2007; Nguyen et al., 2009a), text categorization (Cancedda et al., 2003), word sense disambiguation (Gliozzo et al., 2005) and semantic role labeling (SRL), e.g. (Kazama and Torisawa, 2005; Che et al., 2006a; Moschitti et al., 2008).

However, ours is the first study on the use of several combinations of kernels applied to several structures on very complex data from the Jeopardy! domain.

7 Final Remarks and Conclusion

In this paper we have experimented with advanced structural kernels applied to several kinds of syntactic/semantic linguistic structures for the classification of questions in a new application domain, i.e. Jeopardy!. Our findings are summarized hereafter:

[11]Available at `http://cogcomp.cs.illinois.edu/Data/QA/QC/`

First, it should be noted that basic kernels, such as STK, PTK and SK, when applied to new representations, i.e. syntactic/semantic structures, constitute new kernels. Thus structural representations play a major role and, from this perspective, our paper makes a significant contribution.

Second, the experimental results show that the higher variability of Jeopardy! questions prevents us from achieving generalization with typical syntactic patterns even if they are derived by powerful methods such as STK. In contrast, partial patterns, such as those provided by PTK applied to constituency (or dependency) trees, prove to be effective.

In particular, STK has been considered as the best kernel for exploiting syntactic information in constituency trees, e.g. it is state-of-the-art in: QC (Zhang and Lee, 2003; Moschitti et al., 2007; Moschitti, 2008); SRL, (Moschitti et al., 2008; Moschitti et al., 2005; Che et al., 2006b); pronominal coreference resolution (Yang et al., 2006; Versley et al., 2008) and Relation Extraction (Zhang et al., 2006; Nguyen et al., 2009b). We showed that, in the complex domain of Jeopardy!, STK surprisingly provides low accuracy whereas PTK is rather effective and greatly outperforms STK. We have also provided an explanation of such behavior by means of error analysis: in contrast with traditional question classification, which focuses on basic syntactic patterns (e.g. "what", "where", "who" and "how"). Figure 5 shows that PTK captures partial patterns that are important for more complex questions like those in Jeopardy!

Third, we derived other interesting findings for NLP related to this novel domain, e.g.: (i) the impact of dependency trees is similar to the one of constituency trees. (ii) A simple computational representation of shallow semantics, i.e. PASS (Moschitti, 2008), does not work in Jeopardy!. (iii) Sequence kernels on category cues, i.e., higher level of lexical semantics, improve question classification. (iv) RBC jointly used with statistical approaches is helpful to tackle the Jeopardy! complexity.

Next, our kernel models improve up to 20 absolute percent points over n-grams based approaches, reaching a significant accuracy of about 70%. Watson, exploiting such a classifier, improved previous versions using RBC and no definition classification both in definition-only evaluations and in game-based evaluations.

Finally, we point out that:

- Jeopardy! has a variety of different special question types that are handled differently. We focus on kernel methods for definition question for two reasons. First, their recognition relies heavily on parse structures and is therefore more amenable to the approach proposed in the paper than the recognition of other question types. Second, definition is by far the most frequent special question type in Jeopardy!; therefore, we can obtain sufficient data for training and testing.

- We were unable to address the whole QC problem using a statistical model due to the lack of sufficient training data for most special question classes. Furthermore, we focused only on the definition classification and its impact on system performance due to space reasons.

- Our RBC has a rather imbalanced trade-off between Precision and Recall. This may not be the best operating point, but the optimal point is difficult to obtain empirically for an RBC, which is a strong motivation of the work in this paper. We experimented with tuning the trade-off between Precision and Recall with the RBC, but since RBC uses hand-crafted rules and does not have a parameter for that, ultimately the statistical approach proved more effective.

In future work, we plan to extend the current research by investigating models capable of exploiting predicate argument structures for question classification and answer reranking. The use of syntactic/semantic kernels is a promising research direction (Basili et al., 2005; Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b). In this perspective kernel learning is a very interesting research line, considering the complexity of representation and classification problems in which our kernels operate.

Acknowledgements

This work has been supported by the IBM's Open Collaboration Research (OCR) awards program. We are deeply in debt with Richard Johansson, who produced the earlier syntactic/semantic representations of the Jeopardy! questions from the text format.

References

Kisuh Ahn, Johan Bos, Stephen Clark, James R. Curran, Tiphaine Dalmas, Jochen L. Leidner, Matthew B. Smillie, and Bonnie Webber. 2004. Question answering with qed and wee at trec-2004. In E. M. Voorhees and L. P. Buckland, editors, *The Thirteenth Text REtrieval Conference, TREC 2004*, pages 595–599, Gaitersburg, MD.

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of CoNLL-2005*, pages 1–8, Ann Arbor, Michigan. Association for Computational Linguistics.

M. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of ACM SIGIR*.

S. Blair-Goldensohn, K. R. McKeown, and A. H. Schlaikjer. 2004. Answering definitional questions: A hybrid approach. In M. Maybury, editor, *Proceedings of AAAI 2004*. AAAI Press.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In *Proceedings of ECIR 2007, Rome, Italy*.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *In Proceedings of CIKM '07*.

Phil Blunsom, Krystle Kocik, and James R. Curran. 2006. Question classification with log-linear models. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 615–616, New York, NY, USA. ACM.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT and EMNLP*, pages 724–731, Vancouver, British Columbia, Canada, October.

Razvan C. Bunescu. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of ACL*.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*.

Wanxiang Che, Min Zhang, Ting Liu, and Sheng Li. 2006a. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 73–80, Sydney, Australia, July. Association for Computational Linguistics.

Wanxiang Che, Min Zhang, Ting Liu, and Sheng Li. 2006b. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 73–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Y. Chen, M. Zhou, and S. Wang. 2006. Reranking answers from definitional QA using language models. In *Proceedings of ACL*.

Charles Clarke, Gordon Cormack, and Thomas Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th SIGIR Conference*, pages 358–365.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL'02*.

H. Cui, M. Kan, and T. Chua. 2005. Generic soft pattern models for definitional QA. In *Proceedings of SIGIR*, Salvador, Brazil. ACM.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*, pages 423–429, Barcelona, Spain, July.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Hal Daumé III and Daniel Marcu. 2004. Np bracketing by maximum entropy tagging and SVM reranking. In *Proceedings of EMNLP'04*.

A. Echihabi and D. Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of ACL*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3).

Danilo Giampiccolo, Pamela Froner, Anselmo Peñas, Christelle Ayache, Dan Cristea, Valentin Jijkoun, Petya Osenova, Paulo Rocha, Bogdan Sacaleanu, and Richard Suteliffe. 2007. Overview of the CLEF 2007 multilingual question anwering track. In *Proceedings of the Cross Language Evaluation Forum*.

Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of ACL'05*, pages 403–410.

A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. 2006. Question answering with lcc chaucer at trec 2006. In *Proceedings of TREC*.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. *Advances in Kernel Methods – Support Vector Learning*, 13.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 183–187, Manchester, United Kingdom.

Michael Kaisser and Bonnie Webber. 2007. Question answering based on semantic roles. In *Proceedings of the Workshop on Deep Linguistic Processing*, DeepLP '07, pages 41–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jun'ichi Kazama and Kentaro Torisawa. 2005. Speeding up training with tree kernels for node relation labeling. In *Proceedings of HLT-EMNLP'05*.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

Shyong Lam, David Pennock, Dan Cosley, and Steve Lawrence. 2003. 1 billion pages = 1 milllion dollars? mining the web to pay "who wants to be a millionaire? In *Proceedings of the 19th Conference on Uncertainty in AI*.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of ACL*.

Michael C. McCord. 1980. Slot grammars. *Computational Linguistics*.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York, Albany.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, United States.

Alessandro Moschitti, Bonaventura Coppola, Ana-Maria Giuglea, and Roberto Basili. 2005. Hierarchical semantic role labeling. In *CoNLL 2005 shared task*.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of ECML'06*, pages 318–329.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *Proceeding of CIKM '08*, NY, USA.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009a. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of EMNLP*, pages 1378–1387, Singapore, August.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009b. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387, Morristown, NJ, USA. Association for Computational Linguistics.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Yutaka Sasaki, Chuan-Jie Lin, Kuang-hua Chen, and Hsin-Hsi Chen. 2007. Overview of the NTCIR-6 cross-lingual question answering (CLQA) task. In *Proceedings of the 6th NTCIR Workshop on Evaluation of Information Access Technologies*.

Y. Sasaki. 2005. Question answering as question-biased term extraction: A new approach toward multilingual qa. In *Proceedings of ACL*, pages 215–222.

John Shawe-Taylor and Nello Cristianini. 2004. *LaTeX User's Guide and Document Reference Manual*. Kernel Methods for Pattern Analysis, Cambridge University Press.

D. Shen and M. Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*.

L. Shen, A. Sarkar, and A. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Proceedings of EMNLP*, Sapporo, Japan.

S. Small, T. Strzalkowski, T. Liu, S. Ryan, R. Salkin, N. Shimizu, P. Kantor, D. Kelly, and N. Wacholder. 2004. Hitiqa: Towards analytical question answering. In *Proceedings of COLING*.

M. Surdeanu, M. Ciaramita, and H. Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of ACL-HLT*, Columbus, Ohio.

J. Suzuki, Y. Sasaki, and E. Maeda. 2002. Svm answer selection for open-domain question answering. In *Proceedings of Coling*, pages 974–980.

Jun Suzuki, Hirotoshi Taira, Yutaka Sasaki, and Eisaku Maeda. 2003. Question classification using hdag kernel. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, pages 61–68, Sapporo, Japan, July. Association for Computational Linguistics.

Ivan Titov and James Henderson. 2006. Porting statistical parsers with data-defined kernels. In *Proceedings of CoNLL-X*.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008. Coreference systems based on kernels methods. In *The 22nd International Conference on Computational Linguistics (Coling'08)*, Manchester, England.

Ellen M. Voorhees and Hoa Trang Dang. 2006. Overview of the TREC 2005 question answering track. In *Proceedings of the TREC 2005 Conference*.

E. M. Voorhees. 2004. Overview of the trec 2004 question answering track. In *Proceedings of TREC 2004*.

Xiaofeng Yang, Jian Su, and Chewlim Tan. 2006. Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. COLING-ACL 06*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of EMNLP-ACL*, pages 181–201.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM Press.

Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering relations between named entities from a large raw corpus using tree similarity-based clustering. In *Proceedings of IJCNLP'2005, Lecture Notes in Computer Science (LNCS 3651)*, pages 378–389, Jeju Island, South Korea.

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring Syntactic Features for Relation Extraction using a Convolution tree kernel. In *Proceedings of NAACL*.

Multiword Expression Identification with Tree Substitution Grammars: A Parsing *tour de force* with French

Spence Green[*], **Marie-Catherine de Marneffe**[†], **John Bauer**[*], **and Christopher D. Manning**[*†]
[*]Computer Science Department, Stanford University
[†]Linguistics Department, Stanford University
{spenceg,mcdm,horatio,manning}@stanford.edu

Abstract

Multiword expressions (MWE), a known nuisance for both linguistics and NLP, blur the lines between syntax and semantics. Previous work on MWE identification has relied primarily on surface statistics, which perform poorly for longer MWEs and cannot model discontinuous expressions. To address these problems, we show that even the simplest parsing models can effectively identify MWEs of arbitrary length, and that Tree Substitution Grammars achieve the best results. Our experiments show a 36.4% F1 absolute improvement for French over an n-gram surface statistics baseline, currently the predominant method for MWE identification. Our models are useful for several NLP tasks in which MWE pre-grouping has improved accuracy.

1 Introduction

Multiword expressions (MWE) have long been a challenge for linguistic theory and NLP. There is no universally accepted definition of the term, but MWEs can be characterized as "idiosyncratic interpretations that cross word boundaries (or spaces)" (Sag et al., 2002) such as *traffic light*, or as "frequently occurring phrasal units which are subject to a certain level of semantic opaqueness, or non-compositionality" (Rayson et al., 2010).

MWEs are often opaque fixed expressions, although the degree to which they are fixed can vary. Some MWEs do not allow morphosyntactic variation or internal modification (e.g., *in short*, but **in shorter* or **in very short*). Other MWEs are "semi-fixed," meaning that they can be inflected or undergo internal modification. The type of modification is often limited, but not predictable, so it is not possible to enumerate all variants (Table 1).

French				English			
à			terme	in the		near	term
à		court	terme	in the		short	term
à	très	court	terme	in the	very	short	term
à		moyen	terme	in the		medium	term
à		long	terme	in the		long	term
à	très	long	terme	in the	very	long	term

Table 1: Semi-fixed MWEs in French and English. The French adverb *à terme* 'in the end' can be modified by a small set of adjectives, and in turn some of these adjectives can be modified by an adverb such as *très* 'very'. Similar restrictions appear in English.

Merging known MWEs into single tokens has been shown to improve accuracy for a variety of NLP tasks: dependency parsing (Nivre and Nilsson, 2004), constituency parsing (Arun and Keller, 2005), sentence generation (Hogan et al., 2007), and machine translation (Carpuat and Diab, 2010). Most experiments use gold MWE pre-grouping or language-specific resources like WordNet. For unlabeled text, the best MWE identification methods, which are based on surface statistics (Pecina, 2010), suffer from sparsity induced by longer n-grams (Ramisch et al., 2010). A dilemma thus exists: MWE knowledge is useful, but MWEs are hard to identify.

In this paper, we show the effectiveness of statistical parsers for MWE identification. Specifically, Tree Substitution Grammars (TSG) can achieve a 36.4% F1 absolute improvement over a state-of-the-art surface statistics method. We choose French, which has pervasive MWEs, for our experiments. Parsing models naturally accommodate discontinuous MWEs like phrasal verbs, and provide syntactic subcategorization. By contrast, surface statistics methods are usually limited to binary judgements for contiguous n-grams or dependency bigrams.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 725–735,
Edinburgh, Scotland, UK, July 27–31, 2011.

	FTB (train)	WSJ (train)
Sentences	13,449	39,832
Tokens	398,248	950,028
#Word Types	28,842	44,389
#Tag Types	30	45
#Phrasal Types	24	27
Per Sentence		
Depth (μ/σ^2)	4.03 / 0.360	4.18 / 0.730
Breadth (μ/σ^2)	13.5 / 6.79	10.7 / 4.59
Length (μ/σ^2)	29.6 / 17.3	23.9 / 11.2
Constituents (μ)	20.3	19.6
μ Constituents / μ Length	0.686	0.820

Table 2: Gross corpus statistics for the pre-processed FTB (training set) and WSJ (sec. 2-21). The FTB sentences are longer with broader syntactic trees. The FTB POS tag set has 33% fewer types than the WSJ. The FTB dev set OOV rate is 17.77% vs. 12.78% for the WSJ.

Type		#Total	#Single	%Single	%Total
MWN	*noun*	9,680	2,737	28.3	49.7
MWADV	*adverb*	3,852	449	11.7	19.8
MWP	*prep.*	3,526	342	9.70	18.1
MWC	*conj.*	814	73	8.97	4.18
MWV	*verb.*	585	243	41.5	3.01
MWD	*det.*	328	69	21.0	1.69
MWA	*adj.*	324	126	38.9	1.66
MWPRO	*pron.*	266	33	12.4	1.37
MWCL	*clitic*	59	1	1.69	0.30
MWET	*foreign*	24	18	0.75	0.12
MWI	*interj.*	4	2	0.50	0.02
		19,462	4,093	21.0%	100.0%

Table 3: Frequency distribution of the 11 MWE subcategories in the FTB (training set). MWEs account for 7.08% of the bracketings and 13.0% of the tokens in the treebank. Only 21% of the MWEs occur once ("single").

We first introduce a new instantiation of the French Treebank that, unlike previous work, does not use gold MWE pre-grouping. Consequently, our experimental results also provide a better baseline for parsing raw French text.

2 French Treebank Setup

The corpus used in our experiments is the French Treebank (Abeillé et al. (2003), version from June 2010, hereafter FTB). In French, there is a linguistic tradition of lexicography which compiles lists of MWEs occurring in the language. For example, Gross (1986) shows that dictionaries contain about 1,500 single-word adverbs but that French contains over 5,000 multiword adverbs. MWEs occur in every part-of-speech (POS) category (e.g., noun *trousse de secours* 'first-aid kit'; verb *faire main-basse* [do hand-low] 'seize'; adverb *comme dans du beurre* [as in butter] 'easily'; adjective *'à part entière'* 'wholly').

The FTB explicitly annotates MWEs (also called *compounds* in prior work). We used the subset of the corpus with functional annotations, not for those annotations but because this subset is known to be more consistently annotated. POS tags for MWEs are given not only at the MWE level, but also internally: most tokens that constitute an MWE also have a POS tag. Table 2 compares this part of the FTB to the WSJ portion of the Penn Treebank.

2.1 Preprocessing

The FTB requires significant pre-processing prior to parsing.

Tokenization We changed the default tokenization for numbers by fusing adjacent digit tokens. For example, *500 000* is tagged as an MWE composed of two words *500* and *000*. We made this *500000* and retained the MWE POS, although we did not mark the new token as an MWE. For consistency, we used one token for punctuated numbers like "17,9".

MWE Tagging We marked MWEs with a flat bracketing in which the phrasal label is the MWE-level POS tag with an "MW" prefix, and the preterminals are the internal POS tags for each terminal. The resulting POS sequences are not always unique to MWEs: they appear in abundance elsewhere in the corpus. However, some MWEs contain normally ungrammatical POS sequences (e.g., adverb *à la va vite* 'in a hurry': P D V ADV [at the goes quick]), and some words appear only as part of an MWE, such as *insu* in *à l'insu de* 'to the ignorance of'.

Labels We augmented the basic FTB label set—which contains 14 POS tags and 19 phrasal tags—in two ways. First, we added 16 finer-grained POS tags for punctuation.[1] Second, we added the 11 MWE

[1] Punctuation tag clusters—as used in the WSJ—did not improve accuracy. Enriched tag sets like that of Crabbé and Candito (2008) could also be investigated and compared to our results since Evalb is insensitive to POS tags.

labels shown in Table 3, resulting in 24 total phrasal categories.

Corrections Historically, the FTB suffered from annotation errors such as missing POS and phrasal tags (Arun and Keller, 2005). We found that this problem has been largely resolved in the current release. However, 1,949 tokens and 36 MWE spans still lacked tags. We restored the labels by first assigning each token its most frequent POS tag elsewhere in the treebank, and then assigning the most frequent MWE phrasal category for the resulting POS sequence.[2]

Split We used the 80/10/10 split described by Crabbé and Candito (2008). However, they used a previous release of the treebank with 12,531 trees. 3,391 trees have been added to the present version. We appended these extra trees to the training set, thus retaining the same development and test sets.

2.2 Comparison to Prior FTB Representations

Our pre-processing approach is simple and automatic[3] unlike the three major instantiations of the FTB that have been used in previous work:

Arun-Cont and Arun-Exp (Arun and Keller, 2005): Two instantiations of the full 20,000 sentence treebank that differed principally in their treatment of MWEs: (1) Cont, in which the tokens of each MWE were concatenated into a single token (*en moyenne* $\rightarrow$ *en_moyenne*); (2) Exp, in which they were marked with a flat structure. For both representations, they also gave results in which coordinated phrase structures were flattened. In the published experiments, they mistakenly removed half of the corpus, believing that the multi-terminal (per POS tag) annotations of MWEs were XML errors (Schluter and Genabith, 2007).

MFT (Schluter and Genabith, 2007): Manual revision to 3,800 sentences. Major changes included coordination raising, an expanded POS tag set, and the correction of annotation errors. Like Arun-Cont, MFT contains concatenated MWEs.

FTB-UC (Candito and Crabbé, 2009): An instantiation of the functionally annotated section that makes a distinction between MWEs that are "syntactically regular" and those that are not. Syntactically regular MWEs were given internal structure, while all other MWEs were concatenated into single tokens. For example, nouns followed by adjectives, such as *loi agraire* 'land law' or *Union monétaire et économique* 'monetary and economic Union' were considered syntactically regular. They are MWEs because the choice of adjective is arbitrary (*loi agraire* and not **loi agricole*, similarly to 'coal black' but not *'crow black' for example), but their syntactic structure is not intrinsic to MWEs. In such cases, FTB-UC gives the MWE a conventional analysis of an NP with internal structure. Such analysis is indeed sufficient to recover the meaning of these semantically compositional MWEs that are extremely productive. On the other hand, the FTB-UC loses information about MWEs with non-compositional semantics.

Almost all work on the FTB has followed Arun-Cont and used gold MWE pre-grouping. As a result, most results for French parsing are analogous to early results for Chinese, which used gold word segmentation, and Arabic, which used gold clitic segmentation. Candito et al. (2010) were the first to acknowledge and address this issue, but they still used FTB-UC (with some pre-grouped MWEs). Since the syntax and definition of MWEs is a contentious issue, we take a more agnostic view—which is consistent with that of the FTB annotators—and leave them tokenized. This permits a data-oriented approach to MWE identification that is more robust to changes to the status of specific MWE instances.

To set a baseline prior to grammar development, we trained the Stanford parser (Klein and Manning, 2003) with no grammar features, achieving 74.2% labeled F1 on the development set (sentences ≤ 40 words). This is lower than the most recent results obtained by Seddah (2010). However, the results are not comparable: the data split was different, they made use of morphological information, and more importantly they concatenated MWEs. The focus of

[2] 73 of the unlabeled word types did not appear elsewhere in the treebank. All but 11 of these were nouns. We manually assigned the correct tags, but we would not expect a negative effect by deterministically labeling all of them as nouns.

[3] We automate tree manipulation with Tregex/Tsurgeon (Levy and Andrew, 2006). Our pre-processing package is available at http://nlp.stanford.edu/software/lex-parser.shtml.

our work is on models and data representations that enable MWE identification.

3 MWEs in Lexicon-Grammar

The MWE representation in the FTB is close to the one proposed in the Lexicon-Grammar (Gross, 1986). In the Lexicon-Grammar, MWEs are classified according to their global POS tags (noun, verb, adverb, adjective), and described in terms of the sequence of the POS tags of the words that constitute the MWE (e.g., "N de N" *garde d'enfant* [guard of child] 'daycare', *pied de guerre* [foot of war] 'at the ready'). In other words, MWEs are represented by a flat structure. The Lexicon-Grammar distinguishes between units that are fixed and have to appear as is (*en tout et pour tout* [in all and for all] 'in total') and units that accept some syntactic variation such as admitting the insertion of an adverb or adjective, or the variation of one of the words in the expression (e.g., a possessive as in 'from the top of one's hat'). It also notes whether the MWE displays some selectional preferences (e.g., it has to be preceded by a verb or by an adjective).

Our FTB instantiation is largely consistent with the Lexicon-Grammar. Recall that we defined different MWE categories based on the global POS. We now detail three of the categories.

MWN The MWN category consists of proper nouns (1a), foreign common nouns (1b), as well as common nouns. The common nouns appear in several syntactically regular sequences of POS tags (2). Multiword nouns allow inflection (singular vs. plural) but no insertion.

(1) a. *London Sunday Times, Los Angeles*
b. *week - end, mea culpa, joint - venture*

(2) a. N A: *corps médical* 'medical staff', *dette publique* 'public debt'
b. N P N: *mode d'emploi* 'instruction manual'
c. N N: *numéro deux* 'number two', *maison mère* [house mother] 'headquarters', *grève surprise* 'sudden strike'
d. N P D N: *impôt sur le revenu* 'income tax', *ministre de l'économie* 'finance minister'

MWA Multiword adjectives appear with different POS sequences (3). They include numbers such as *vingt et unième* '21st'. Some items in (3b) allow internal variation: some adverbs or adjectives can be added to both examples given (*à <u>très</u> haut risque*, *de <u>toute</u> dernière minute*).

(3) a. P N: *d'antan* [from before] 'old', *en question* 'under discussion'
b. P A N: *à haut risque* 'high-risk', *de dernière minute* [from the last minute] 'at the eleventh hour'
c. A C A: *pur et simple* [pure and simple] 'straightforward', *noir et blanc* 'black and white'

MWV Multiword verbs also appear in several POS sequences (4). All verbs allow number and tense inflections. Some MWVs containing a noun or an adjective allow the insertion of a modifier (e.g., *donner grande satisfication* 'give great satisfaction'), whereas others do not. When an adverb intervenes between the main verb and its complement, the FTB marks the two parts of the MWV discontinuously (e.g., [MWV [V prennent]] [ADV déjà] [MWV [P en] [N cause]] 'already take into account').

(4) a. V N: *avoir lieu* 'take place', *donner satisfaction* 'give satisfaction'
b. V P N: *mettre en place* 'put in place', *entrer en vigueur* 'to come into effect'
c. V P ADV: *mettre à mal* [put at bad] 'harm', *être à même* [be at same] 'be able'
d. V D N P N: *tirer la sonnette d'alarme* 'ring the alarm bell', *avoir le vent en poupe* 'to have the wind astern'

4 Parsing Models

We develop two parsers for French with the goal of improving MWE identification. The first is a manually-annotated grammar that we incorporate into the Stanford parser. Manual annotation results in human interpretable grammars that can inform future treebank annotation decisions. Moreover, the grammar can be used as the base distribution in our second model, a Probabilistic Tree Substitution Grammar (PTSG) parser. PTSGs learn parameters for tree

Feature	States	Tags	F1	ΔF1
—	4325	31	74.21	
tagPA	4509	215	76.94	+2.73
markInf	4510	216	77.42	+0.48
markPart	4511	217	77.73	+0.31
markVN	5986	217	78.32	+0.59
markCoord	7361	217	78.45	+0.13
markDe	7521	233	79.11	+0.66
markP	7523	235	79.34	+0.23
markMWE	7867	235	79.23	−0.11

Table 4: Effects on grammar size and labeled F1 for each of the manual state splits (development set, sentences $\leq$ 40 words). *markMWE* decreases overall accuracy, but increases both the number of correctly parsed trees (by 0.30%) and per category MWE accuracy.

fragments larger than basic CFG rules. PTSG rules may also be lexicalized. This means that commonly observed collocations—some of which are MWEs—can be stored in the grammar.

4.1 Stanford Parser

We configure the Stanford parser with settings that are effective for other languages: selective parent annotation, lexicon smoothing, and factored parsing. We use the head-finding rules of Dybro-Johansen (2004), which we find to yield an approximately 1.0% F1 development set improvement over those of Arun (2004). Finally, we include a simple unknown word model consisting entirely of surface features:

- Nominal, adjectival, verbal, adverbial, and plural suffixes
- Contains a digit or punctuation
- Is capitalized (except the first word in a sentence)
- Consists entirely of capital letters
- If none of the above, add a one- or two-character suffix

Combined with the grammar features, this unknown word model yields 97.3% tagging accuracy on the development set.

4.1.1 Grammar Development

Table 4 lists the symbol refinements used in our grammar. Most of the features are POS splits as many phrasal tag splits did not lead to any improvement. Parent annotation of POS tags (*tagPA*) captures information about the external context. *markInf* and *markPart* accomplish a finite/nonfinite distinction: they respectively specify whether the verb is an infinitive or a participle based on the type of the grandparent node. *markVN* captures the notion of verbal distance as in Klein and Manning (2003).

We opted to keep the COORD phrasal tag, and to capture parallelism in coordination, we mark COORD with the type of its child (NP, AP, VPinf, etc.). *markDe* identifies the preposition *de* and its variants (*du*, *des*, *d'*) which is very frequent and appears in several different contexts. *markP* identifies prepositions which introduce PPs modifying a noun. Marking other kinds of prepositional modifiers (e.g., verb) did not help. *markMWE* adds an annotation to several MWE categories for frequently occuring POS sequences. For example, we mark MWNs that occur more than 600 times (e.g., "N P N" and "N N").

4.2 DP-TSG Parser

A shortcoming of CFG-based grammars is that they do not explicitly capture idiomatic usage. For example, consider the two utterances:

(5) a. He [$_{\text{MWV}}$ kicked the bucket] .
b. He [$_{\text{VP}}$ kicked [$_{\text{NP}}$ the pail]] .

The examples in (5) may be equally probable and receive the same analysis under a PCFG; words are generated independently. However, recall that in our representation, (5a) should receive a flat analysis as MWV, whereas (5b) should have a conventional analysis of the verb *kicked* and its two arguments.

An alternate view of parsing is one in which new utterances are built from previously observed fragments. This is the original motivation for data oriented parsing (DOP) (Bod, 1992), in which "idiomaticity is the rule rather than the exception" (Scha, 1990). If we have seen the collocation *kicked the bucket* several times before, we should store that whole fragment for later use.

We consider a variant of the non-parametric PTSG model of Cohn et al. (2009) in which tree fragments are drawn from a Dirichlet process (DP) prior.[4] The DP-TSG can be viewed as a DOP model with Bayesian parameter estimation. A PTSG is a 5-tuple $\langle V, \Sigma, R, \diamondsuit, \boldsymbol{\theta} \rangle$ where $c \in V$ are non-terminals;

[4]Similar models were developed independently by O'Donnell et al. (2009) and Post and Gildea (2009).

α_c	DP concentration parameter for each $c \in V$
$P_0(e\|c)$	CFG base distribution
$\boldsymbol{x}$	Set of non-terminal nodes in the treebank
$\mathcal{S}$	Set of sampling sites (one for each $x \in \boldsymbol{x}$)
S	A block of sampling sites, where $S \subseteq \mathcal{S}$
$\boldsymbol{b} = \{b_s\}_{s \in \mathcal{S}}$	Binary variables to be sampled ($b_s = 1 \rightarrow$ frontier node)
$\boldsymbol{z}$	Latent state of the segmented treebank
m	Number of sites $s \in S$ s.t. $b_S = 1$
$\boldsymbol{n} = \{n_{c,e}\}$	Sufficient statistics of $\boldsymbol{z}$
$\Delta n^{S:m}$	Change in counts by setting m sites in S

Table 5: DP-TSG model notation. For consistency, we largely follow the notation of Liang et al. (2010). Note that $\boldsymbol{z} = (\boldsymbol{b}, \boldsymbol{x})$, and as such $z = \langle c, e \rangle$.

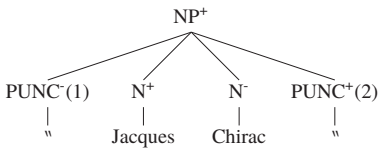


Figure 1: Example of two conflicting sites of the same type. Define the *type* of a site $t(\boldsymbol{z}, s) \stackrel{\text{def}}{=} (\Delta n^{s:0}, \Delta n^{s:1})$. Sites (1) and (2) above have the same type since $t(\boldsymbol{z}, s_1) = t(\boldsymbol{z}, s_2)$. However, the two sites *conflict* since the probabilities of setting b_{s_1} and b_{s_2} both depend on counts for the tree fragment rooted at NP. Consequently, sites (1) and (2) are not exchangeable: the probabilities of their assignments depend on the order in which they are sampled.

$t \in \Sigma$ are terminals; $e \in R$ are elementary trees;[5] $\diamondsuit \in V$ is a unique start symbol; and $\theta_{c,e} \in \boldsymbol{\theta}$ are parameters for each tree fragment. A PTSG derivation is created by successively applying the substitution operator to the leftmost *frontier node* (denoted by c^+). All other nodes are *internal* (denoted by c^-).

In the supervised setting, DP-TSG grammar extraction reduces to a segmentation problem. We have a treebank T that we segment into the set R, a process that we model with Bayes' rule:

$$p(R \mid T) \propto p(T \mid R)\, p(R) \tag{1}$$

Since the tree fragments completely specify each tree, $p(T \mid R)$ is either 0 or 1, so all work is performed by the prior over the set of elementary trees.

The DP-TSG contains a DP prior for each $c \in V$ (Table 5 defines further notation). We generate $\langle c, e \rangle$ tuples as follows:

$$\theta_c | c, \alpha_c, P_0(\cdot|c) \sim DP(\alpha_c, P_0)$$
$$e | \theta_c \sim \theta_c$$

The data likelihood is given by the latent state $\boldsymbol{z}$ and the parameters $\boldsymbol{\theta}$: $p(\boldsymbol{z}|\boldsymbol{\theta}) = \prod_{z \in \boldsymbol{z}} \theta_{c,e}^{n_{c,e}(\boldsymbol{z})}$. Integrating out the parameters, we have:

$$p(\boldsymbol{z}) = \prod_{c \in V} \frac{\prod_e (\alpha_c P_0(e|c))^{\overline{n_{c,e}(\boldsymbol{z})}}}{\alpha_c^{\overline{n_{c,\cdot}(\boldsymbol{z})}}} \tag{2}$$

where $x^{\overline{n}} = x(x+1)\ldots(x+n-1)$ is the rising factorial. (§A.1 contains ancillary details.)

Base Distribution The base distribution P_0 is the same maximum likelihood PCFG used in the Stanford parser.[6,7] After applying the manual state splits, we perform simple right binarization, collapse unary rules, and replace rare words with their signatures (Petrov et al., 2006).

For each non-terminal type c, we learn a stop probability $s_c \sim \text{Beta}(1, 1)$. Under P_0, the probability of generating a rule $A^+ \rightarrow B^-\ C^+$ composed of non-terminals is

$$P_0(A^+ \rightarrow B^-\ C^+) = p_{\text{MLE}}(A \rightarrow B\ C) s_B (1 - s_C) \tag{3}$$

For lexical insertion rules, we add a penalty proportional to the frequency of the lexical item:

$$P_0(c \rightarrow t) = p_{\text{MLE}}(c \rightarrow t) p(t) \tag{4}$$

where $p(t)$ is equal to the MLE unigram probability of t in the treebank. Lexicalizing a rule makes it very specific, so we generally want to avoid lexicalization with rare words. Empirically, we found that this penalty reduces overfitting.

Type-based Inference Algorithm To learn the parameters $\boldsymbol{\theta}$ we use the collapsed, block Gibbs sampler of Liang et al. (2010). We sample binary variables b_s associated with each non-terminal node/site in the treebank. The key idea is to select a block of exchangeable sites S of the same *type* that do not *conflict* (Figure 1). Since the sites in S are exchangeable, we can set $\boldsymbol{b}_S$ randomly so long as we know m, the number of sites with $b_s = 1$. Because this algorithm is a not a contribution of this paper, we refer the reader to Liang et al. (2010).

[5] We use the terms *tree fragment* and *elementary tree* interchangeably.

[6] The Stanford parser is a product model, so the results in §5.1 include the contribution of a dependency parser.

[7] Bansal and Klein (2010) also experimented with symbol refinement in an all-fragments (parametric) TSG for English.

After each Gibbs iteration, we sample each s_c directly using binomial-Beta conjugacy. We re-sample the DP concentration parameters α_c with the auxiliary variable procedure of West (1995).

Decoding We compute the rule score of each tree fragment from a single grammar sample as follows:

$$\theta_{c,e} = \frac{n_{c,e}(\boldsymbol{z}) + \alpha_c P_0(e|c)}{n_{c,\cdot}(\boldsymbol{z}) + \alpha_c} \quad (5)$$

To make the grammar more robust, we also include all CFG rules in P_0 with zero counts in $\boldsymbol{n}$. Scores for these rules follow from (5) with $n_{c,e}(\boldsymbol{z}) = 0$.

For decoding, we note that the derivations of a TSG are a CFG parse forest (Vijay-Shanker and Weir, 1993). As such, we can use a Synchronous Context Free Grammar (SCFG) to translate the 1-best parse to its derivation. Consider a unique tree fragment e_i rooted at X with frontier γ, which is a sequence of terminals and non-terminals. We encode this fragment as an SCFG rule of the form

$$[X \rightarrow \gamma \ , \ X \rightarrow i, Y_1, \ldots, Y_n] \quad (6)$$

where $Y_1, \ldots, Y_n$ is the sequence of non-terminal nodes in γ.[8] During decoding, the input is rewritten as a sequence of tree fragment (rule) indices $\{i, j, k, \ldots\}$. Because the TSG substitution operator always applies to the leftmost frontier node, we can deterministically recover the monolingual parse with top-down re-writes of $\diamondsuit$.

The SCFG formulation has a practical benefit: we can take advantage of the heavily-optimized SCFG decoders for machine translation. We use `cdec` (Dyer et al., 2010) to recover the Viterbi derivation under a DP-TSG grammar sample.

5 Experiments

5.1 Standard Parsing Experiments

We evaluate parsing accuracy of the Stanford and DP-TSG models (Table 6). For comparison, we also include the Berkeley parser (Petrov et al., 2006).[9] For the DP-TSG, we initialized all b_s with fair coin tosses and ran for 400 iterations, after which likelihood stopped improving.

[8] This formulation is due to Chris Dyer.

[9] Training settings: right binarization, no parent annotation, six split-merge cycles, and random initialization.

	Leaf Ancestor		Evalb			
	Corpus	Sent	LP	LR	F1	EX%
PA-PCFG	0.793	0.812	68.1	67.0	67.6	10.5
DP-TSG	0.823	0.842	75.6	76.0	75.8	15.1
Stanford	0.843	0.861	77.8	79.0	78.4	17.5
Berkeley	**0.880**	**0.891**	**82.4**	**82.0**	**82.2**	**21.4**

Table 6: Standard parsing experiments (test set, sentences $\leq$ 40 words). All parsers exceed 96% tagging accuracy. Berkeley and DP-TSG results are the average of three independent runs.

We report two different parsing metrics. *Evalb* is the standard labeled precision/recall metric.[10] *Leaf Ancestor* measures the cost of transforming guess trees to the reference (Sampson and Babarczy, 2003). It was developed in response to the non-terminal/terminal ratio bias of Evalb, which penalizes flat treebanks like the FTB. The range of the score is between 0 and 1 (higher is better). We report micro-averaged (whole corpus) and macro-averaged (per sentence) scores.

In terms of parsing accuracy, the Berkeley parser exceeds both Stanford and DP-TSG. This is consistent with previous experiments for French by Seddah et al. (2009), who show that the Berkeley parser outperforms other models. It also matches the ordering for English (Cohn et al., 2010; Liang et al., 2010). However, the standard baseline for TSG models is a simple parent-annotated PCFG (PA-PCFG). For English, Liang et al. (2010) showed that a similar DP-TSG improved over PA-PCFG by 4.2% F1. For French, our gain is a more substantial 8.2% F1.

5.2 MWE Identification Experiments

Table 7 lists overall and per-category MWE identification results for the parsing models. Although DP-TSG is less accurate as a general parsing model, it is more effective at identifying MWEs.

The predominant approach to MWE identification is the combination of lexical association measures (surface statistics) with a binary classifier (Pecina, 2010). A state-of-the-art, language independent package that implements this approach for higher order n-grams is `mwetoolkit` (Ramisch et al., 2010).[11] In Table 8 we compare DP-TSG to both

[10] Available at http://nlp.cs.nyu.edu/evalb/ (v.20080701).

[11] Available at http://multiword.sourceforge.net/. See §A.2 for

	#gold	Stanford	DP-TSG	Berkeley
MWET	3	0.0	0.0	0.0
MWV	26	64.0	57.7	50.7
MWA	8	26.1	32.2	29.8
MWN	456	64.1	67.6	67.1
MWD	15	70.3	65.5	70.1
MWPRO	17	73.7	78.0	76.2
MWADV	220	74.6	72.7	70.4
MWP	162	81.3	80.5	77.7
MWC	47	83.5	83.5	80.8
	954	70.1	**71.1**	69.6

Table 7: MWE identification per category and overall results (test set, sentences ≤ 40 words). MWI and MWCL do not occur in the test set.

Model	F1
`mwetoolkit` All	15.4
PA-PCFG	32.6
`mwetoolkit` Filter	34.7
PA-PCFG+Features	63.1
DP-TSG	**71.1**

Table 8: MWE identification F1 of the best parsing model vs. the `mwetoolkit` baseline (test set, sentences ≤ 40 words). PA-PCFG+Features includes the grammar features in Table 4, which is the CFG from which the TSG is extracted. For `mwetoolkit`, *All* indicates the inclusion of all n-grams in the training corpus. *Filter* indicates prefiltering of the training corpus by removing rare n-grams (see §A.2 for details).

`mwetoolkit` and the CFG from the which the TSG is extracted. The TSG-based parsing model outperforms `mwetoolkit` by 36.4% F1 while providing syntactic subcategory information.

6 Discussion

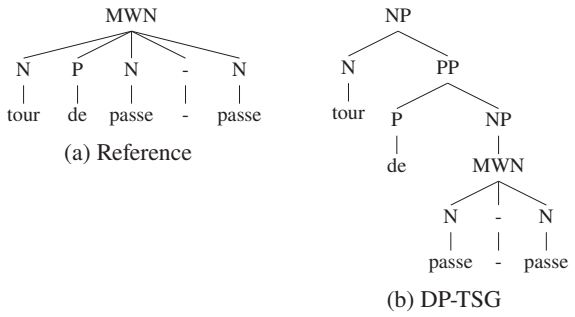
Automatic learning methods run the risk of producing uninterpretable models. However, the DP-TSG model learns useful generalizations over MWEs. A sample of the rules is given in Table 9. Some specific sequences like "[MWN [coup de N]]" are part of the grammar: such rules can indeed generate quite a few MWEs, e.g., *coup de pied* 'kick', *coup de coeur*, *coup de foudre* 'love at first sight', *coup de main* 'help', *coup d'état*, *coup de grâce* (note that only some of these MWEs are seen in the training data). For MWV, "V de N" as in *avoir de cesse* 'give no peace', *perdre de vue* [lose from sight] 'forget', *prendre de vitesse* [take from speed] 'outpace'), is learned. For prepositions, the grammar stores full subtrees of MWPs, but can also generalize the structure of very frequent sequences: "en N de" occurs in many multiword prepositions (e.g., *en compagnie de*, *en face de*, *en matière de*, *en terme de*, *en cours de*, *en faveur de*, *en raison de*, *en fonction de*). The TSG grammar thus provides a categorization of MWEs consistent with the Lexicon-Grammar. It also learns verbal phrases which contain discontinuous MWVs due to the insertion of an adverb or negation such as "[VN [MWV va] [MWADV d'ailleurs] [MWV bon train]]" [go indeed well], "[VN [MWV a] [ADV jamais] [MWV été question d']]" [has never been in question].

MWN
sociétés de N
prix de N
coup de N
N d'état
N de N
N à N

MWV
sous - V
faire N
V les moyens
V de N
V en N

MWP
de l'ordre de
y compris
au N de
en N de
ADV de

Table 9: Sample of the TSG rules learned.



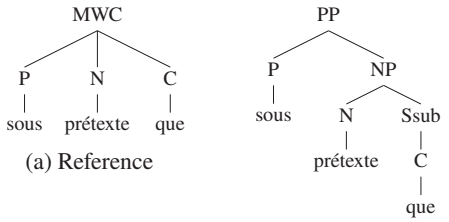
(a) Reference (b) DP-TSG

Figure 2: Example of an MWE error for *tour de passe-passe* 'magic trick'. (dev set)

A significant fraction of errors for MWNs occur with adjectives that are not recognized as part of the MWE. For example, since *établissements privés* 'private corporation' is unseen in the training data, it is not found. Sometimes the parser did not recognize the whole structure of an MWE. Figure 2 shows an example where the parser only found a subpart of the MWN *tour de passe-passe* 'magic trick'.

Other DP-TSG errors are due to inconsistencies in the FTB annotation. For example, *sous prétexte que*

configuration details.



(a) Reference

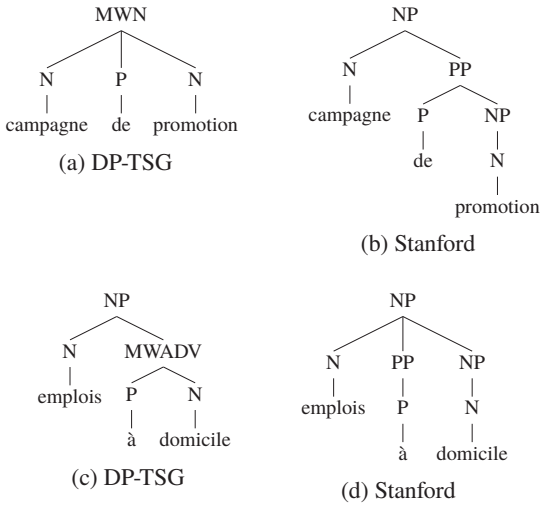
(b) Reference

Figure 3: Example of an inconsistent FTB annotation for *sous prétexte que* 'on the pretext of'.

'on the pretext of' is tagged as both MWC and as a regular PP structure (Figure 3). However, the parser always assigns a MWC structure, which is a better analysis than the gold annotation. We expect that more consistent annotation would help the DP-TSG more than the CFG-based parsers.

The DP-TSG is not immune to false positives: in *Le marché national, fait-on remarquer, est enfin en régression* ... 'The national economy, people at last note, is going down' the parser tags *marché national* as MWN. As noted, the boundary of what should and should not count as an MWE can be fuzzy, and it is therefore hard to assess whether or not this should be an MWE. The FTB does not mark it as one.

There are multiple examples were the DP-TSG found the MWE whereas Stanford (its base distribution) did not, such as in Figure 4. Note that the "N P N" structure is quite frequent for MWNs, but the TSG correctly identifies the MWADV in *emplois à domicile* [jobs at home] 'homeworking'.



(a) DP-TSG

(b) Stanford

(c) DP-TSG

(d) Stanford

Figure 4: Correct analyses by DP-TSG. (dev set)

7 Related Work

There is a voluminous literature on MWE identification. Here we review closely related syntax-based methods.[12] The linguistic and computational attractiveness of lexicalized grammars for modeling idiosyncratic constructions in French was identified by Abeillé (1988) and Abeillé and Schabes (1989). They manually developed a small Tree Adjoining Grammar (TAG) of 1,200 elementary trees and 4,000 lexical items that included MWEs. The classic statistical approach to MWE identification, Xtract (Smadja, 1993), used an incremental parser in the third stage of its pipeline to identify predicate-argument relationships. Lin (1999) applied information-theoretic measures to automatically-extracted dependency relations to find MWEs. To our knowledge, Wehrli (2000) was the first to use syntactically annotated corpora to improve a parser for MWE identification. He proposed to rank analyses of a symbolic parser based on the presence of collocations, although details of the ranking function were not provided.

The most similar work to ours is that of Nivre and Nilsson (2004), who converted a Swedish corpus into two versions: one in which MWEs were left as tokens, and one in which they were merged. On the first version, they showed that a deterministic dependency parser could identify MWEs at 71.1% F1, albeit without subcategory information. On the second version—which simulated perfect MWE identification—they showed that labeled attachment improved by about 1%.

Recent statistical parsing work on French has included Stochastic Tree Insertion Grammars (STIGs), which are related to TAGs, but with a restricted adjunction operation.[13] Seddah et al. (2009) and Seddah (2010) showed that STIGs underperform CFG-based parsers on the FTB. In their experiments, MWEs were concatenated.

[12] See Seretan (2011) for a comprehensive survey of syntax-based methods for MWE identification. For an overview of n-gram methods like `mwetoolkit`, see Pecina (2010).

[13] TSGs differ from TAGs and STIGs in that they do not include an adjunction operator.

8 Conclusion

The main result of this paper is that an existing statistical parser can achieve a 36.4% F1 absolute improvement for MWE identification over a state-of-the-art n-gram surface statistics package. Parsers also provide syntactic subcategorization, and do not require pre-filtering of the training data. We have also demonstrated that TSGs can capture idiomatic usage better than a PCFG. While the DP-TSG, which is a relatively new parsing model, still lags state-of-the-art parsers in terms of overall labeling accuracy, we have shown that it is already very effective for other tasks like MWE identification. We plan to improve the DP-TSG by experimenting with alternate parsing objectives (Cohn et al., 2010), lexical representations, and parameterizations of the base distribution. A particularly promising base distribution is the latent variable PCFG learned by the Berkeley parser. However, initial experiments with this distribution were negative, so we leave further development to future work.

We chose French for these experiments due to the pervasiveness of MWEs and the availability of an annotated corpus. However, MWE lists and syntactic treebanks exist for many of the world's major languages. We will investigate automatic conversion of these treebanks (by flattening MWE bracketings) for MWE identification.

A Appendix

A.1 Notes on the Rising Factorial

The rising factorial—also known as the ascending factorial or Pochhammer symbol—arises in the context of samples from a Dirichlet process (see Prop. 3 of Antoniak (1974) for details). For a positive integer n and a complex number x, the rising factorial $x^{\overline{n}}$ is defined[14] by

$$\begin{aligned} x^{\overline{n}} &= x(x+1)\dots(x+n-1) \\ &= \prod_{j=1}^{n}(x+j-1) \end{aligned} \tag{7}$$

The rising factorial can be generalized to a complex number α with the gamma function:

$$x^{\overline{\alpha}} = \frac{\Gamma(x+\alpha)}{\Gamma(x)} \tag{8}$$

where $x^{\overline{0}} \equiv 1$.

In our type-based sampler, we computed (7) directly in a dynamic program. We found that (8) was prohibitively slow for sampling.

A.2 `mwetoolkit` Configuration

We configured `mwetoolkit`[15] with the four standard lexical features: the maximum likelihood estimator, Dice's coefficient, pointwise mutual information (PMI), and Student's t-score. We added the POS sequence for each n-gram as a single feature. We removed the web counts features to make the experiments comparable. To compensate for the absence of web counts, we computed the lexical features using the gold lemmas from the FTB instead of using an automatic lemmatizer.

Since MWE n-grams only account for a small fraction of the n-grams in the corpus, we filtered the training and test sets by removing all n-grams that occurred once. To further balance the proportion of MWEs, we trained on all valid MWEs plus 10x randomly selected non-MWE n-grams. This proportion matches the fraction of MWE/non-MWE tokens in the FTB. Since we generated a random training set, we reported the average of three independent runs.

We created feature vectors for the training n-grams and trained a binary Support Vector Machine (SVM) classifier with Weka (Hall et al., 2009). Although `mwetoolkit` defaults to a linear kernel, we achieved higher accuracy on the development set with an RBF kernel.

The FTB is sufficiently large for the corpus-based methods implemented in `mwetoolkit`. Ramisch et al. (2010)'s experiments were on Genia, which contains 18k sentences and 490k tokens, similar to the FTB. Their test set had 895 sentences, smaller than ours. They reported 30.6% F1 for their task against an Xtract baseline, which only obtained 7.3% F1. These results are comparable in magnitude to our FTB results.

Acknowledgments We thank Marie Candito, Chris Dyer, Dan Flickinger, Percy Liang, Carlos Ramisch, Djamé Seddah, and Val Spitkovsky for their helpful comments. The first author is supported by a National Defense Science and Engineering Graduate (NDSEG) fellowship.

[14] We adopt the notation of Knuth (1992).

[15] We re-implemented `mwetoolkit` in Java for compatibility with Weka and our pre-processing routines.

References

A. Abeillé and Y. Schabes. 1989. Parsing idioms in lexicalized TAGs. In *EACL*.

A. Abeillé, L. Clément, and A. Kinyon, 2003. *Building a treebank for French*, chapter 10. Kluwer.

A. Abeillé. 1988. Parsing French with Tree Adjoining Grammar: some linguistic accounts. In *COLING*.

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174.

A. Arun and F. Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of French. In *ACL*.

A. Arun. 2004. Statistical parsing of the French treebank. Technical report, University of Edinburgh.

M. Bansal and D. Klein. 2010. Simple, accurate parsing with an all-fragments grammar. In *ACL*.

R. Bod. 1992. A computation model of language performance: Data-Oriented Parsing. In *COLING*.

M. Candito and B. Crabbé. 2009. Improving generative statistical parsing with semi-supervised word clustering. In *IWPT*.

M. Candito, B. Crabbé, and P. Denis. 2010. Statistical French dependency parsing: treebank conversion and first results. In *LREC*.

M. Carpuat and M. Diab. 2010. Task-based evaluation of multiword expressions: a pilot study in statistical machine translation. In *HLT-NAACL*.

T. Cohn, S. Goldwater, and P. Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *HLT-NAACL*.

T. Cohn, P. Blunsom, and S. Goldwater. 2010. Inducing tree-substitution grammars. *JMLR*, 11:3053–3096, Nov.

B. Crabbé and M. Candito. 2008. Expériences d'analyse syntaxique statistique du français. In *TALN*.

A. Dybro-Johansen. 2004. Extraction automatique de grammaires à partir d'un corpus français. Master's thesis, Université Paris 7.

C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, et al. 2010. `cdec`: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *ACL System Demonstrations*.

M. Gross. 1986. Lexicon-Grammar: the representation of compound words. In *COLING*.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explorations Newsletter*, 11:10–18.

D. Hogan, C. Cafferkey, A. Cahill, and J. van Genabith. 2007. Exploiting multi-word units in history-based probabilistic generation. In *EMNLP-CoNLL*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

D. E. Knuth. 1992. Two notes on notation. *American Mathematical Monthly*, 99:403–422, May.

R. Levy and G. Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *LREC*.

P. Liang, M. I. Jordan, and D. Klein. 2010. Type-based MCMC. In *HLT-NAACL*.

D. Lin. 1999. Automatic identification of non-compositional phrases. In *ACL*.

J. Nivre and J. Nilsson. 2004. Multiword units in syntactic parsing. In *Methodologies and Evaluation of Multiword Units in Real-World Applications (MEMURA)*.

T. J. O'Donnell, J. B. Tenenbaum, and N. D. Goodman. 2009. Fragment grammars: Exploring computation and reuse in language. Technical report, MIT Computer Science and Artificial Intelligence Laboratory Technical Report Series, MIT-CSAIL-TR-2009-013.

P. Pecina. 2010. Lexical association measures and collocation extraction. *Language Resources and Evaluation*, 44:137–158.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

M. Post and D. Gildea. 2009. Bayesian learning of a tree substitution grammar. In *ACL-IJCNLP, Short Papers*.

C. Ramisch, A. Villavicencio, and C. Boitet. 2010. `mwetoolkit`: a framework for multiword expression identification. In *LREC*.

P. Rayson, S. Piao, S. Sharoff, S. Evert, and B. Moirón. 2010. Multiword expressions: hard going or plain sailing? *Language Resources and Evaluation*, 44:1–5.

I. A. Sag, T. Baldwin, F. Bond, A. Copestake, and D. Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *CICLing*.

G. Sampson and A. Babarczy. 2003. A test of the leaf-ancestor metric for parse accuracy. *Natural Language Engineering*, 9:365–380.

R. Scha, 1990. *Taaltheorie en taaltechnologie: competence en performance*, pages 7–22. Landelijke Vereniging van Neerlandici (LVVNjaarboek).

N. Schluter and J. Genabith. 2007. Preparing, restructuring, and augmenting a French treebank: Lexicalised parsers or coherent treebanks? In *Pacling*.

D. Seddah, M. Candito, and B. Crabbé. 2009. Cross parser evaluation and tagset variation: a French treebank study. In *IWPT*.

D. Seddah. 2010. Exploring the Spinal-STIG model for parsing French. In *LREC*.

V. Seretan. 2011. *Syntax-Based Collocation Extraction*, volume 44 of *Text, Speech, and Language Technology*. Springer.

F. Smadja. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19:143–177.

K. Vijay-Shanker and D. J. Weir. 1993. The use of shared forests in tree adjoining grammar parsing. In *EACL*.

E. Wehrli. 2000. Parsing and collocations. In *Natural Language Processing—NLP 2000*, volume 1835 of *Lecture Notes in Computer Science*, pages 272–282. Springer.

M. West. 1995. Hyperparameter estimation in Dirichlet process mixture models. Technical report, Duke University.

Modelling Discourse Relations for Arabic

Amal Alsaif
University of Leeds
Leeds, UK
LS2 9JT
amalalsaif@yahoo.co.uk

Katja Markert
University of Leeds
Leeds, UK
LS2 9JT
markert@comp.leeds.ac.uk

Abstract

We present the first algorithms to automatically identify explicit discourse connectives and the relations they signal for Arabic text. First we show that, for Arabic news, most adjacent sentences are connected via explicit connectives in contrast to English, making the treatment of explicit discourse connectives for Arabic highly important. We also show that explicit Arabic discourse connectives are far more ambiguous than English ones, making their treatment challenging. In the second part of the paper, we present supervised algorithms to address automatic discourse connective identification and discourse relation recognition. Our connective identifier based on gold standard syntactic features achieves almost human performance. In addition, an identifier based solely on simple lexical and automatically derived morphological and POS features performs with high reliability, essential for languages that do not have high-quality parsers yet. Our algorithm for recognizing discourse relations performs significantly better than a baseline based on the connective surface string alone and therefore reduces the ambiguity in explicit connective interpretation.

1 Introduction

The automatic detection of discourse relations, such as causal, contrast or temporal relations, is useful for many applications such as automatic summarization (Marcu, 2000), question answering (Girju, 2003), sentiment analysis (Somasundaran et al., 2008) and readability assessment (Pitler and Nenkova, 2008). This task has recently seen renewed interest due to the growing availability of large-scale corpora annotated for discourse relations, such as the Penn Discourse Treebank (Prasad et al., 2008a).

In the Penn Discourse Treebank (PDTB), local discourse relations (also called *senses*) such as CAUSAL or CONTRAST are annotated. They hold between two text segments (so-called *arguments*) that express abstract entities such as events, facts and propositions. Annotated discourse relations can be signalled explicitly by so-called *discourse connectives* (Marcu, 2000; Webber et al., 1999; Prasad et al., 2008a) or hold implicitly between adjacent sentences in the same paragraph, i.e. are not signalled by a specific surface string. In Ex. 1, the connective *while* indicates an explicit CONTRAST between the attitudes of John and Richard. In Ex. 2, the connective *while* indicates an explicit TEMPORAL relation. In Ex. 3, an implicit CAUSAL relation between the first and second sentence holds. We indicate discourse connectives and the two arguments they relate via annotated square brackets.

(1) [John liked adventure,]$_{Arg2}$ [while]$_{DC}$[Richard was cautious]$_{Arg2}$

(2) [The children were crying loudly]$_{Arg1}$[while]$_{DC}$,[their mother was cooking]$_{Arg2}$

(3) [I cannot eat any dessert.]$_{Arg1}$ [I have eaten far too much already.]$_{Arg2}$

Although similar corpora for other languages are being developed such as for Hindi (Prasad et al., 2008b), Turkish (Zeyrek and Webber, 2008), Chinese (Xue, 2005) and, by ourselves, for Arabic (Al-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 736–747,
Edinburgh, Scotland, UK, July 27–31, 2011.

Saif and Markert, 2010), efforts in the automated recognition of discourse connectives, arguments and relations have so far almost exclusively centered on English.

In contrast we present the first models for discourse relations for Arabic, focusing on explicit connectives. This focus is partially justified by the fact that this first study for a new language should center on the superficially more straightforward case and that no annotations for implicit relations are yet available for Arabic. More importantly, however, we make two essential claims (Section 4). Firstly, Arabic discourse connectives are more ambiguous than their English counterparts, i.e cases such as *while* which can signal different relations dependent on context (see Example 1 and 2) are far more frequent. This makes their treatment more *challenging*. Secondly, discourse relations between adjacent sentences in Arabic tend to be expressed via an explicit connective, at least for the news genre, i.e. cases such as Example 3 are rarer. This makes the treatment of explicit connectives *essential*.

We tackle two tasks for explicit Arabic connectives in this paper, which are further discussed in Section 2. Discourse connective recognition needs to distinguish between discourse usage of potential connectives and non-discourse usage (such as the use of *while* as a noun). We show in Section 5 that we can distinguish discourse- and non-discourse usage for potential connectives in Arabic with very high reliability, even without parsed data, a fact that is important for languages with fewer high quality NLP tools available. We then present an algorithm for relation identification in Section 6 that shows small but significant gains over assigning the most frequent relation for each connective. We discuss future work and conclude in Section 7.

2 The Tasks

The handling of explicit connectives can be split into three tasks (Pitler and Nenkova, 2009). The first task of *discourse connective recognition* distinguishes between the *discourse usage and non-discourse usage* of potential connectives. Whereas some potential connectives such as the Arabic connective لكن */lkn/but* almost always have discourse usage, this is not true for all potential connectives.[1] Thus, the discourse usage of Arabic رغبة */rġbh/desire* needs to be distinguished from its use as a noun. Conjunctions such as و */w/and,* او */āw/or* can have discourse usage or just conjoin two non-abstract entities as in عمر و ساره */ʕmr w sārh/Omar and Sarah.*

The second task is *discourse connective interpretation* where a discourse connective in context is assigned a discourse relation. Again, some connectives are largely unambiguous in this respect. For example, لكن */lkn/but* signals almost always a CONTRAST relation. However, there are connectives where this is not the case, such as منذ */mnḏ/since* which has a CAUSAL and a TEMPORAL sense.

The third task is argument identification which identifies the arguments' position and extent. In this paper we tackle Task 1 and Task 2 for Arabic in a supervised machine learning framework.

3 Related work

Annotated Discourse Corpora and Linguistic Background. Discourse relations are widely studied in theoretical linguistics (Halliday and Hasan, 1976; Hobbs, 1985), where also different relation taxonomies have been derived (Hobbs, 1985; Knott and Sanders, 1998; Mann and Thompson, 1988; Marcu, 2000). Different inventories have been used in English corpora annotated for discourse relations (Hobbs, 1985; Prasad et al., 2008a; Carlson et al., 2002) which also differ in other respects (such as whether they prescribe a tree structure for discourse annotation). However, the annotation level of existing Arabic corpora has not yet included the discourse layer, making our work the first to address this problem for Arabic on a larger scale.

Automatic discourse parsing: explicit relations. There is no work on discourse connective recognition, interpretation and argument assignment for Arabic, so that we break entirely new ground here. However, the two tasks we explore (discourse connective recognition and discourse connective disambiguation) have been tackled for English.[2] (Pitler

[1] Arabic examples contain in order: the Arabic right-to-left script, the transliteration (standards ISO/R 233 and DIN 31635) and the English translation (if possible).

[2] There is also substantial work on argument identification (Wellner and Pustejovski, 2007; Elwell and Baldridge, 2008)

and Nenkova, 2009) use gold standard syntactic features as well as the connective surface string in a supervised model for discourse connective recognition. They achieve very high results with this approach. We will (i) show that similar features work well for Arabic (ii) take into account Arabic-specific morphological properties that improve results further and (iii) present a robust version of this approach that does not rely on full parsing or gold standard syntactic annotations.

With regard to discourse connective interpretation, (Miltsakaki et al., 2005) concentrate on disambiguating the three connectives *since, while, when* only, using a very small set of features indicating tense and temporal markers in arguments. They achieve good improvements over a "most frequent relation per connective" baseline. A more comprehensive study on all discourse connectives in the PDTB (Pitler et al., 2008; Pitler and Nenkova, 2009) reveals that most connectives are not ambiguous in English. Using syntactic features of the connective, they achieve only a very small improvement over a "most frequent relation per connective baseline" for which significance tests are not given. We will show that for Arabic, discourse connectives are more highly ambiguous with regard to the relations they convey. We will present a supervised learning model that uses a wider feature set and that achieves small but significant improvements over the most frequent relation per connective baseline.

Automatic discourse parsing: implicit relations. Implicit relations have excited substantial interest for English. This includes work in the framework of RST (Soricut and Marcu, 2003; duVerle and Prendinger, 2009; Marcu and Echihabi, 2002), SDRT (Baldridge and Lascarides, 2005), GraphBank (Wellner et al., 2006), the PDTB (Blair-Goldensohn et al., 2007; Pitler et al., 2009; Lin et al., 2009; Wang et al., 2010; Zhou et al., 2010; Louis and Nenkova, 2010) or framework-independent (Sporleder and Lascarides, 2008).[3] The task is challenging as implicits behave substantially differently from explicits (Sporleder and Lascarides, 2008) and often need world knowledge (Lin et al., 2009). However, features/approaches that have shown improvement over a baseline are word pairs (Sporleder and Lascarides, 2008), production rules and syntactic trees (Wang et al., 2010; Lin et al., 2009) as well as language modelling (Zhou et al., 2010). As we only deal with explicit connectives this work is not directly comparable to ours, although we do explore some of the suggested features for improving explicit connective disambiguation.

4 An Arabic Discourse Corpus

We annotate news articles from the Arabic Penn Treebank (Part 1 v2.0) (Maamouri and Bies, 2004) for explicitly marked discourse relations. This is the first discourse-annotated corpus for Arabic, whose initial development stages we have described in (Al-Saif and Markert, 2010). We summarize this previous work and extend it by including agreement studies for arguments in Sections 4.1 and 4.2. In Sections 4.3, 4.4 and 4.5. we then present a corpus study on the corpus which shows our major claim as to the importance and high levels of ambiguity of Arabic discourse connectives.

4.1 Annotation Principles

We overall follow the annotation principles in the Penn Discourse Treebank for explicit connectives (for example, arguments can occur at any distance from the connectives). The relation set we use is a more coarse-grained version of the PDTB relations with two relations added — BACKGROUND and SIMILARITY — that we found in our Arabic news texts. The final, hierarchically organized, relation set of 17 discourse relations is shown in Fig 1.

Further adaptations necessary for Arabic are the inclusion of clitics as connectives such as ل*/l/for,* ب */b/by,with* and ف */f/then* . In addition, differently to English, prepositions were included as connectives as these are frequently used to express discourse relations in Arabic. In these cases, normally argument 2 is the so-called *Al-Masdar*.[4] Typical examples are وصول */wṣwl/arrival* from the verb وصل */wṣ/to arrive* and محاولة */mḥāwlh/attempt* from the verb حاول

but we do not discuss this work in depth here.

[3] Some work does not make the distinction between implicit and explicit and/or treats them in a joint framework (Soricut and Marcu, 2003; Wellner et al., 2006; Wang et al., 2010).

[4] The medieval Arabic grammar schools, the Basra and Kufa, debated whether the noun (almasdar) or the verb is the most basic element of language (Ryding, 2005).

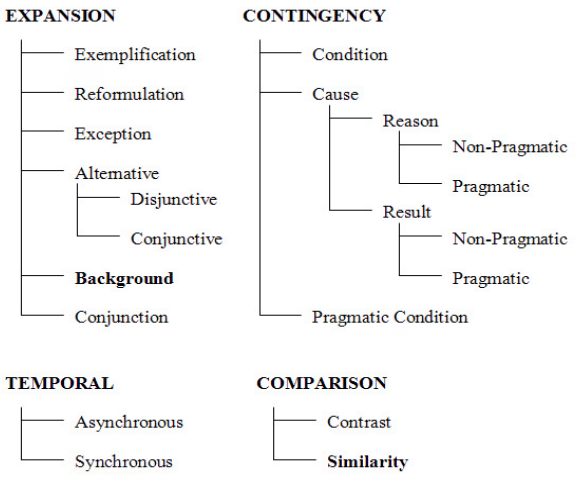


Figure 1: Discourse relations for Arabic

/ḥāwl/to try. Al-Masdar is formed using morphological patterns well-known in the Arabic grammatical tradition: major Arabic grammars list around 60 patterns although some other references also claim that the patterns are many more as well as more unpredictable (Abdl al latif et al., 1997; Wright, 2008; Ryding, 2005). Al-Masdar forms do not fit into one grammatical or morphological category in English: they might correspond to a gerund, a nominalization or a noun which is not a nominalization. Some examples are listed in Table 1.

Table 1: A list of Al-MaSdar patterns, examples and their English correspondence

Root	Pattern	MaSdar	Translation
سبح */sbḥ*	فعالة */fʿalh*	سباحة */sbaḥh*	swimming
نفذ */nfḏ*	تفعيل */tfʿyl*	تنفيذ */tnfyḏ*	execution
دفع */dfʿ*	فعال */fʿāl*	دفاع */dfāʿ*	defence
زرع */zrʿ*	فعالة */fʿālh*	زراعة */zrāʿh*	agriculture
حرب */ḥrb*	فعل */fʿl*	حرب */ḥrb*	war

An example of Al-MaSdar as argument of a discourse relation is Ex. 4, where تبليغ*/tblyġ/informing* is the *Al-MaSdar* form of بلغ*/blġ/inform*.

(4) [ذهبنا الى مركز الشرطة]$_{Arg1}$[ل]$_{DC}$[لتبليغ عن فقدان وثائق الشركة الرسمية]$_{Arg2}$

[ḏhbnā ’lā mrkz al-šrṭt.]$_{Arg1}$[l]$_{DC}$[ltblyġ ʿn fqdān wṯāʾiq alšrkh alrsmyh]$_{Arg2}$

[We went to the police station]$_{Arg1}$ [for]$_{DC}$ [informing about the loss of the company’s official documents.]$_{Arg2}$

4.2 Agreement Studies

The occurrences of a precompiled list of 107 potential discourse connectives were annotated independently by 2 native Arabic speakers on 537 news texts. Agreement was measured for the distinction of discourse vs. non-discourse usage, relation assignment and argument assignment.

Agreement for the classification tasks of discourse connective recognition and relation assignment was measured using kappa (Siegel and Castellan, 1956). Argument agreement was measured by `agr`, a directional measure (Wiebe et al., 2005). It measures the word overlap between the text spans of two judges (ann1 and ann2). `agr(ann1||ann2)` measures the proportion of words ann1 annotated that were also annotated by ann2.

$$agr(ann1||ann2) = \frac{|ann1\ matching\ ann2|}{|ann1|}$$

Discourse connective recognition proved to be highly reliable with percentage agreement of 0.95 and a kappa of 0.88 on the 23,331 occurrences of the 107 potential discourse connectives. 5586 of the potential connectives were agreed on by both annotators to have discourse usage and agreement for relations and argument assignment was measured on these. As shown in Table 2, kappa on all 17 relations was low with 0.57 — it turned out that this was due to the frequent, almost rhetorical use of the connective و */w/and* at the beginning of paragraphs, which is a genre convention for Arabic news that normally does not convey a specific discourse relation. Disregarding such occurrences of و*/w/and*, kappa rises to good agreement: 0.69 for fine-grained relations and 0.75 when measuring agreement between the 4 major relations EXPANSION, CONTINGENCY, COMPARISON and TEMPORAL.

Argument agreement on the 5586 agreed connectives is shown in Table 3. We report high word overlap via `agr` (over 90%) for Arg2, which is the argument syntactically attached to the connective, and lesser but still substantial agreement for Arg1.

Table 2: Inter-annotator reliability for discourse relation assignment

All connectives (5586)	
Observed agreement	0.66
Kappa	0.57
Class level	
Observed agreement	0.8
Kappa	0.67
Connectives excluding و /*w*/*and* at BOP (3500)	
Observed agreement	0.74
Kappa	0.69
Class level	
Observed agreement	0.71
Kappa	0.75

Agreed disc. conn	5586	
	Arg1	Arg2
a) exact match		
exact match =1	2361 (42%)	3803 (68%)
exact match =0	699 (13%)	18 (0.3%)
partial match	2526 (45%)	1765 (32%)
b) agr metric		
`agr(ann1||ann2)`	78%	93%
`agr(ann2||ann1)`	74%	93%
Avr (agr)	76%	93%

Table 3: Inter-annotator reliability for arguments Arg1 and Arg2, using two different measurements (a) exact match (b) `agr`

4.3 Gold standard

We produced a unified gold standard. First, we automatically corrected easily made annotator mistakes. With regard to argument extent, we automatically corrected mistakes such as the erroneous inclusion of punctuation marks at the end of clauses/sentences or not including all obligatory complements in a verb phrase argument. The latter relied on the syntactic annotation in the ATB. Second, with regard to discourse relation assignment, we automatically assigned EXPANSION.CONJUNCTION to all disagreed instances of و /*w*/*and* at BOP.[5] A further disambiguation study is necessary for و /*w*/*and* at BOP, which is beyond the scope of this paper.

Finally, an adjudicator not initially involved in annotation reconciled the remaining disagreements at all levels and included annotations for 5 new potential discourse connective types not in our initial connective list but commented on by the annotators during annotation. 3 news files were removed from the corpus — they contained no actual news reports but just a list of headlines.

The final discourse treebank we use has 6328 annotated explicit connectives in 534 files. 68 connective types were found, rising to 80 connective types if we include all modified forms of a connective as distinct types such as بالرغم من/*bālrġm mn,* رغم ان /*rġm ān* as modified forms of رغم /*rġm/although.*

Most discourse connectives were only annotated with a single relation but 5% were annotated with two or more relations (as also allowed in the PDTB). These statistics are summarised in Table 4.

Files	534
Total tagged tokens	126,046 (125KB)
Sentences	3607
Paragraphs	3312
Discourse connectives (tokens)	6328
Distinct connective (types) including modifed form connectives	68 80
Clitic discourse connectives (tokens)	4779 (76%)
Non-clitic discourse connectives (tokens)	1549 (24%)
Relations types (17 single, 38 combined)	55
Single relations (tokens)	6039 (95%)
Combined relations (tokens)	289 (5%)

Table 4: Statistics of the final gold standard corpus

4.4 Importance of explicitly signalled relations

We compared the number of relations between 2 adjacent sentences that were explicitly signalled in English vs. the ones that were explicitly signalled in Arabic, using the PDTB and our corpus (both containing texts of the news genre). Out of a total 44,470 adjacent sentence pairs in the PDTB, 5355

[5] Other instances of و /*w*/*and* are not treated this way.

(12%) were linked by an explicit connective.[6] In contrast, out of the 3073 adjacent sentence pairs in our corpus, 2140 (70%) were linked by an explicit connective, 948 (30%) were linked via non-*wa* connectives. Thus, for our corpus, modeling of explicit connectives is primary: intrasentential relations tend to be marked by connectives anyway in both English and Arabic, and our corpus shows that this is true for most local intersentential relations as well.

4.5 Ambiguity for Arabic discourse connectives

We investigate the ambiguity of Arabic connectives with regard to their sense at class level (4 relations) as well as the more fine-grained level (all 17 relations). We restrict our investigation to the connective occurrences that were annotated with a single relation (6039 tokens) and also exclude و */w/and* at the beginning of paragraph, leaving 3813 tokens.[7] Of 80 connective types, 52 were unambiguous at the class level and 47 at the fine-grained level. However, many of the most frequent connectives are highly ambiguous. If we just assign the most frequent reading to each of the 3813 connectives, we achieve an accuracy of 82.7% at the class-level and 74.3% at the more fine-grained level for relation assignment, leaving a substantial error margin. This contrasts with the English PDTB, where at the class-level 92% can be achieved with this simple method and 85% at the second-level.[8]

5 Discourse Connective Recognition

We distinguished discourse vs. non-discourse usage for all potential connectives in the 534 gold standard files. As headers and footers in the news files never contained true discourse connectives, we disregarded these, leaving 20,312 potential discourse connectives of which 6328 are actual connectives.

[6]Connections between subclauses or phrases in different, adjacent sentences were included in the count.

[7]We automatically assigned CONJUNCTION to many occurrences of و */w/and* at BOP (Section 4.3) so that it is not sensible to include these occurrences in a study of human-assigned ambiguity.

[8]The second level in the PDTB with its 16 relations corresponds approximately to our fine-grained inventory. This comparison can only be appropriate due to slight differences in the lower-grained relation inventory.

5.1 Features

Apart from the surface string of the potential connective *Conn*, we use the following features. Features are either extracted from raw files tokenized by white space only (M2) or from raw files tokenized by white space and tagged by the Stanford tagger[9] (Models M3, M4) or from the Arabic Treebank (ATB) gold standard part-of-speech and parse annotation (models M5-M9). The syntactic features (Syn) are inspired by (Pitler and Nenkova, 2009). Lexical/POS patterns of surrounding words, clitic features and Al-Masdar are novel.

Surface Features (SConn). These include the position of the potential connective (sentence-initial, medial or final). The *type* of the potential connective is `Simple` when the potential connective is a single token not attached to other tokens, `PotClitic` when it is attached. Potential connectives containing more than one token have `MoreThanToken` type.

Models where we use ATB or automated tagging (M3-M9) distinguish further between potential clitics that are assigned a POS and ones that are not. Models that use ATB annotation also distinguish between potential connectives that correspond to a phrase in the ATB (`MorethanToken_Phrase`) and the ones that do not (`MorethanToken_NonPhrase`).

Lexical features of surrounding words (Lex). We encode the surface strings of the three words before and after the connective, recording position. These features are especially useful for languages where no accurate parser or tagger is available as lexical patterns can capture discourse and non-discourse usage. For instance, if a potential connective is followed by ان */ān/* it most likely has a discourse function (see Ex. 5).

(5) ان الاطفال يمكن [ان يصابوا بالارهاق]$_{Arg1}$[و]$_{DC}$ [ان يشعروا بالنعاس]$_{Arg2}$ خلال الدراسة اذا لم يناموا جيدا

[*ān ālāṭfāl ymkn [ān yṣābwā bālārhāq*]$_{Arg1}$[*w*]$_{DC}$[*ān yšʕrwā bālnʕās*]$_{Arg2}$ *ẖlāl āldrāsh āḏā lm ynāmwā ğydā*

[Children might be tired]$_{Arg1}$ [and]$_{DC}$ [feel sleepy]$_{Arg2}$ during school time if they did not sleep well

[9]`http://nlp.stanford.edu/software/tagger.shtml`

Part of Speech features (POS). We include the pos tag of the potential connective via the ATB/Stanford Tagger. For potential connectives that consist of more than one token, we combined its ordered POS tags. Thus, the potential connective في حال */fy ḥāl/in case* with its tags (fy PREP)(Hal NOUN)) will receive the pos PREP#NOUN. If a potential connective does not receive a separate POS tag in the ATB/tagger, the value "NONE" is assigned. This allows to distinguish clitics from letters at the start of a word. We also record the POS of the three words before/after the connective (ATB/Stanford Tagger). Similar to lexical patterns, these can capture discourse and non-discourse usage. For instance, if a potential connective is soon followed by a modal, it is more likely to have a discourse function.

Syntactic category of related phrases (Syn). We record the syntactic category of the parent of the potential connective in ATB. For example, it is rare that cases where the parent of the potential connective is an adjective phrase, correspond to discourse-usage. A typical example of a non-discourse usage of و*/w/and* (المدرسة كبيرة و جميلة */ālmdrsh kbyrh w ğmylh/ the school is very large and beautiful*) illustrates this. Unlike English, parents in Arabic often are noun phrases as nominalisations are frequent arguments of prepositional connectives. We also encode the Left sibling category and right sibling category of the connective. For discourse connectives, the right sibling is normally S, SBAR, VP or an NP (if the connective is a preposition).

Al-Masdar feature. Potential connectives followed by Al-Masdar are more likely to have discourse usage (see Section 4.1). Especially prepositions with discourse usage are normally attached to Al-masdar such as in لمحادثة */lmḥādṯh/for contacting* or باجراء */bāğrāʔ/by processing*. Al-Masdar information is not included in the ATB so we constructed a binary Al-Masdar feature from (tagged) text by examining the first noun after the potential connective. We developed an algorithm to judge such a noun as Al-Masdar or not. This algorithm uses a stemmer for Arabic and then determines whether the stem is al-Masdar by a combination of surface-based rules to check whether the stem corresponds to one of the known Al-Masdar patterns.

5.2 Results and Discussion

We used the implementation JRip of the rule-based classifier Ripper in the machine learning tool WEKA with its default settings. We used 10-fold cross-validation throughout. Significance tests are reported using the McNemar test at the significance level of 1%. A most frequent category baseline would assign all potential connectives as *not connective*, achieving an accuracy of 68.9% as only 6328 of our potential 20,312 connectives actually have discourse usage. We built several models using different features. The results are shown in Table 5.

A simple model M1 that only uses the connective string improves significantly over the baseline with 75.7% accuracy but a kappa of only 0.48, showing that this is not a reliable strategy. Models M2-M4 do not rely on gold standard annotation or parsing (in contrast to the models for English in (Pitler and Nenkova, 2009)). Using only surface and lexical features that can be extracted from white-space tokenized raw files in addition to the connective string (M2), gains a substantial improvement over using the connective string alone. This is further improved by using POS tags of connectives and surrounding words with an automatic tagger (M3) and by including the Al-Masdar feature (M4), thus making good use of the morphological properties of Arabic. All differences are statistically significant (M1 $<$ M2 $<$ M3 $<$ M4). The final model is reliable (kappa 0.70), an encouraging result given the absence of parsing and important for resource-scarce languages.

With ATB gold standard tokenisation, tagging and parsing, our models (not surprisingly) improve further showing the same pattern of (M1 $<$ M5 $<$ M6 $<$ M7) with all differences being significant. The final best model achieves highly reliable results (accuracy 92.4%, kappa 0.82). We also conclude that syntactic features are more useful than lexical patterns as model M8 (syntax with no lexical patterns) achieves equally good results as M6. Our models also manage to generalise well over individual connectives. If we leave out the connective string (M9), we still achieve a highly reliable result.

6 Discourse Relation Recognition

When disambiguating the relation that discourse connectives signal, we assume that the arguments of

	Features	Acurr	K
	Baseline (not conn)	68.9	0
M1	Conn only	75.7	0.48
Tokenization by white space + auto tagger			
M2	Conn+ SConn+Lex	85.6	0.62
M3	Conn+ SConn+Lex+POS	87.6	0.69
M4	Conn+SConn+Lex+POS+Masdar	88.5	0.70
ATB-based features			
M5	Conn+SConn+Lex	86.2	0.65
M6	Conn+SConn+Lex+Syn/POS	91.2	0.79
M7	Conn+SConn+Lex+Syn/POS+Masdar	92.4	0.82
M8	Conn+SConn+Syn	91.2	0.79
M9	SConn+Lex+Syn+Masdar	91.2	0.79

Table 5: Performance of different models for identifying discourse connectives.

the connective are known. This is well-established for PDTB relation recognition (Wang et al., 2010; Lin et al., 2009; Miltsakaki et al., 2005). Our models predict single relations on two datasets: (i) all instances of connectives signalling single relations (Set `All`, 6039 instances) (2) all instances apart from the connective و */w/and* at beginning of paragraph as they are affected by the auto-correction process (Set `no-wa-atBOP`, 3813 instances). We use 10-fold cross-validation and JRip as well as a McNemar test at the 5% level for significance tests.

6.1 Features

Whereas some of the features we use have been used for English implicit relation recognition (Lin et al., 2009; Wang et al., 2010; Pitler et al., 2009) , they are new for Arabic and not widely used for explicit connectives. All features are extracted from the ATB gold standard parses.

Connective features. This includes the connective string *Conn*. In addition, we also use the surface connective features and POS of connective described in Section 5. We also use the syntactic path to the connective as a novel feature.

Words and POS of arguments. The words and pos tags of the first three words in Arg1 and Arg2 are used to catch patterns in arguments. For example, when the first word of Arg2 is قد */qd/might/may* or كان */kān/had*, the relation is likely to be EXPANSION.BACKGROUND or EXPANSION.CONJUNCTION. We also measure word overlap between the arguments, hoping to catch relations such as COMPARISON.SIMILARITY.

Masdar. This feature states whether the first or second word in Arg 2 is an Al-Masdar. Many prepositional connectives followed by an Al-Masdar indicate a CONTINGENCY.CAUSE relation (see Ex. 4)

Tense and Negation. Each argument is assigned its tense as one of *perfect, imperfect, future or none*. We also indicate whether the tense of Arg1 or 2 are the same and whether a negation is part of Arg 1 or 2. Inspired by (Miltsakaki et al., 2005), we stipulate that tense is useful for recognizing temporal and causal relations. For example, the arguments of the relation TEMPORAL.SYNCHRONOUS are likely to have the same tense. In contrast, arg1_tense is more likely to be prior to arg2_tense for TEMPORAL.ASYNCHRONOUS and CAUSE relations.

Length, Distance and Order Features. We use the length of arguments (in words), word distance between a connective and its arguments (-1; for arguments in order Arg1_Conn_Arg2_Arg1), tree distance of connective and arguments (0 if connective and an argument are in the same tree) and a binary feature of whether Arg1 and Arg2 are in different sentences. A nominal feature encodes one of the three orders Arg1_Conn_Arg2, Conn_Arg2_Arg1 and Arg1_Conn_Arg2_Arg1, the latter being frequent in Arabic for TEMPORAL.ASYNCHRONOUS relations.

Argument Parent. We record the syntactic parent of each Argument. However, not every argu-

ment corresponds to a complete tree in the ATB — in these cases we extract the category of the parent shared by the first and last word in the argument.

Production Rules. We use all non-lexical production rules that occur more than 10 times in the arguments as binary features. This was inspired by (Lin et al., 2009) who use production rules to good effect for implicit relations in English.

6.2 Results

Table 6 shows the results for fine-grained (17 relations) classification. The baseline of assigning the most frequent relation EXPANSION.CONJUNCTION to every connective performs with an accuracy of 52.5% on Set `All` and 35% on set `no-wa-atBOP`. If we use a model that relies on the discourse connective alone (M1) we achieve results of 77.2%/74.3%, respectively. As noted in Section 4.5 this is substantially lower than what the same model can achieve for English. Including connective and argument features (apart from production rules) in M2, leads to a small but significant improvement.[10] Further incorporation of production rules does not improve the results (M3). In Table 7, we show the results at the class-level (4 relations). Here using additional features over the connective string does not lead to significant improvements.

6.3 Discussion and Error Analysis

We concentrate our discussion on fine-grained classification excluding wa at BOP.

Our improvements in M2 over the connective-only classifier (M1) are in two main areas. First, our model performs generalisation, i.e. outputs some rules that do not use the connective string at all. These achieve a somewhat surprising improvement of M2 over M1 for *u*nambiguous connectives which are too rare to classify via the connective string. In those cases, they either (i) have not been seen in the training data before and are therefore not classifiable when seen first time in the test set or (ii) have been seen in the training data too rarely for the rule-based classifier to develop a rule judged to be more reliable than the default EXPANSION.CONJUNCTION classification. Our data includes 47 unambiguous connective types, accounting for 574 of the 3813 tokens. 30 of these 47 types are so rare that we found mistakes in the connective-only classification, including الا */ālā*/except (2), عقب */ʿqb*(2),طالما */ṭā-lmā*(2), برغم*/brġm*(1). For 14 of these 30 connectives, model M2 was able to use generalised rules to improve relation assignment.[11] These rules involve mainly connective surface and POS features. Thus, sentence-start adverbials consisting of more than one token such as بيد ان */byd ān*(6) and غير ان */ġyr ān*(6) were correctly classified as CONTRAST.

Ref	Features	Acc	K
All connectives (6039)			
	Baseline (CONJUNCTION)	52.5	0
M1	Conn only (1)	77.2	0.60
M2	Conn+Conn_f+ Arg_f (37)	78.8	0.66
M3	Conn+Conn_f+ Arg_f+ Production rules (1237)	78.3	0.65
excluding wa at BOP (3813)			
	Baseline (CONJUNCTION)	35	0
M1	Conn only (1)	74.3	0.65
M2	Conn+Conn_f+ Arg_f (37)	77	0.69
M3	Conn+Conn_f+ Arg_f+ Production rules (1237)	76.7	0.69

Table 6: Performance of different models for identifying fine-grained discourse relations on two datasets.

Ref	Features	Acc	K
All connectives (6039)			
	Baseline (EXPANSION)	62.4	0
M1	Conn only (1)	88.7	0.78
M2	Conn+Conn_f+ Arg_f (37)	88.7	0.78
excluding wa at BOP (3813)			
	Baseline (EXPANSION)	41.8	0
M1	Conn only (1)	82.7	0.74
M2	Conn+Conn_f+ Arg_f (37)	83.5	0.75

Table 7: Performance of different models for identifying class-level discourse relations on two datasets.

[10] Our corpus includes some texts on similar topics where some sentences are (almost) repeated in different texts. To investigate whether our improvements are due to this repetition, we also performed an experiment excluding all repeated instances of feature vectors from the corpus. The results are almost the same and, most importantly, M2 again improves significantly over M1.

[11] For the other 16 connectives neither of the models was able to classify them correctly.

This advantage of our model over the connective-only model might disappear if in a larger corpus more instances of those connectives are found and are still unambiguous. Therefore, we are more interested in how our classifier does on truly ambiguous connectives (33 connective types accounting for 3239 tokens of 3813 overall tokens). We conducted a separate significance test on ambiguous connectives only and found that M2 improves over M1 classification significantly at the 1% level. How well we do on individual connectives depends on their frequency and on their level of ambiguity. If connectives are ambiguous and of low frequency (لو /*lw*, انما /*ānmā*, حال /*ḥāl*/) both M1 and M2 do perform badly on them. If connectives are frequent (10 or more occurrences) and have relatively low ambiguity (majority reading accounts for more than 70% of instances), the overall performance of M1 and M2 with regard to accuracy is also similar, often both using just the connective string. On the other hand, if connectives are frequent and have high ambiguity (i.e. no such clear majority reading), then M2 normally improves (often substantially) on M1. Examples of such connectives are كما /*kmā*, فيما /*fymā*and اثر /*āṯr*. Most of the successful rules use tense in some form, either via part of speech of verbs or via comparing the tense in the two arguments. This, for example, led to a successful recognition of all 9 instances of `Similarity` in the connective kmA (whose majority relation is `Expansion.Conjunction` in 40 out of 65 occurrences). The connective ف */f/then* is distinguished into EXPANSION.EXEMPLIFICATION, CONTINGENCY.CAUSE.RESULT and CONTINGENCY.CAUSE.REASON readings, depending on the lexemes around it, the parents of its arguments, and whether its argument 2 is tensed or not. Thus, nontensed arguments are most often nominalisations leading to a reason reading, whereas a verb phrase as argument 2 and a sentence as argument 1 often is a result reading. However, it is worth reporting that in cases of very high ambiguity, M2 is still far from perfect such as for connectives f ف /*f*/and اثر /*āṯr*.

Some improvements again come from generalised rules: there are some very high-coverage and high precision generalised rules that reduce dependency on the connective string. For example, clitic prepositions (such as ل /*l*/for) can without any further information be clearly classified as `Contingency.Cause.Reason.NonPragmatic` covering 494 occurrences with only 26 mistakes. These are cases where the following argument is normally Al-Masdar.

Our analysis leads us to the following strategy for follow-on work. First of all, a larger corpus is necessary to get more examples for low frequency connectives. Secondly, experiments with different classifiers are worthwhile to conduct to see how our improvements generalise. Third, the most mileage is in further improvements on frequent, ambiguous connectives such as ف /*f*, منذ /*mnḏ*and او /*āw*. This can be achieved with, on the one hand, training connective-specific classifiers on larger data sets but will, on the other hand, also need a wider feature base. From our corpus study, we think that lexico-semantic features such as word pairs and semantic classes of verbal/nominalised arguments are the most promising.

7 Conclusions and Future Work

We have presented the first study on the automatic detection and disambiguation of Arabic discourse connectives. A corpus study showed that these are highly frequent and more ambiguous than their English counterparts. Our automatic algorithms achieve very good results on discourse connective identification, using Arabic morphological properties to good effect. It is particular promising that we do not need parsed data to identify discourse usage of potential connectives reliably. Our algorithm for discourse connective interpretation beats the challenging baseline of assigning the most frequent relation per connective. In future, we will explore further features for connective disambiguation as well as connective-specific classification, combined with semi-supervised algorithms to alleviate data sparseness. We will also develop algorithms for argument identification.

Acknowledgments

Amal Al-Saif is supported by a PhD scholarship from the Imam Muhammad Ibn Saud University, Saudi Arabia. We thank the British Academy for additional funding for the annotation study via Grant SG51944. Also thank you to the 3 anonymous reviewers for their comments.

References

M. Abdl al latif, A. Umar, and M. Zahran. 1997. *Alnhw AlAsAsi*. Dar Alfker Al-Arabi, Cairo, Egypt.

A. AlSaif and K. Markert. 2010. The leeds arabic discourse treebank: Annotating discourse connectives for arabic. In *Language Resources and Evaluation Conference (LREC)*.

J. Baldridge and A. Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proc. Of Conll 2005*.

S. Blair-Goldensohn, K McKeown, and O. Rambow. 2007. Building and refining rhetorical-semantic relation models. In *Proc. of HLT-NAACL 2007*.

L. Carlson, D. Marcu, and M. Okurewski. 2002. Rst discourse treebank. Linguistic Data Consortium.

D. duVerle and H. Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proc. of ACL 2009*.

R. Elwell and J. Baldridge. 2008. Discourse connective argument identification with connective specific rankers. In *Proc. of the International Conference on Semantic Computing*.

R. Girju. 2003. Automatic detection of causal relations for questions answering. In *Proc. of the ACL 2003 Workshop on Multilingual Summarisation and Question Answering*, pages 76–83.

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

J.R. Hobbs. 1985. *On the coherence and structure of discourse*. Center for the Study of Language and Information, Stanford, Calif.

A. Knott and T. Sanders. 1998. The classification of coherence relations and their linguistic markers: An exploration of two languages. *Journal of Pragmatics*, 30(2):135–175.

Z. Lin, M. Kan, and H.T. Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proc. of EMNLP 2009*, pages 343–351.

A. Louis and A. Nenkova. 2010. Creating local coherence: An empirical assessment. In *Proc. of NAACL 2010*.

M. Maamouri and A. Bies. 2004. Developing an Arabic treebank: Methods, guidelines, procedures, and tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages (COLING), Geneva*.

W.C. Mann and S.A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

D. Marcu and A. Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proc. of ACL 2002*.

D. Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT Press.

E. Miltsakaki, N. Dinesh, R. Prasad, A. Joshi, and B. Webber. 2005. Experiments on sense annotation and sense disambiguation of discourse connectives. In *Proc. of the Workshop on Treebanks and Linguistic Theories*.

E. Pitler and A. Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proc. of EMNLP 2008*, pages 186–195.

E. Pitler and A. Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives. In *Proc of ACL-IJCNLP 2009 (Short Papers)*, pages 13–16.

E. Pitler, M. Raghupathy, H. Mehta, A. Nenkova, A. Lee, and A. Joshi. 2008. Easily identifiable discourse relations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008), Manchester, UK, August*.

E. Pitler, A. Louis, and A. Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proc. of ACL-IJCNLP 2009*, pages 683–691.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. 2008a. The Penn discourse treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

R. Prasad, S. Husain, D.M. Sharma, and A. Joshi. 2008b. Towards an Annotated Corpus of Discourse Relations in Hindi. In *The Third International Joint Conference on Natural Language Processing*, pages 7–12. Citeseer.

K.C. Ryding. 2005. *A reference grammar of modern standard Arabic*. Cambridge Univ Pr.

S. Siegel and N.J. Castellan. 1956. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill New York.

S. Somasundaran, J. Wiebe, and J. Ruppenhofer. 2008. Discourse-level opinion interpretation. In *Proc. of Coling 2008*.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proc of HLT-NAACL 2003*.

C. Sporleder and A. Lascarides. 2008. Using automatically labelled examples to classify rhetorical relations: An assessment. *Natural Language Engineering*, 14:369–416.

W. Wang, J. Su, and C. Tan. 2010. Kernel-based discourse relation recognition with temporal ordering information. In *Proc. of ACL 2010*, pages 710–719.

B. Webber, A. Knott, M. Stone, and A. Joshi. 1999. Discourse relations: A structural and presuppositional account using lexicalised TAG. In *Proceedings of*

the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, page 48. Association for Computational Linguistics.

B. Wellner and J. Pustejovski. 2007. Automatically identifying the arguments of discourse connectives. In *Proc. of EMNLP 2007*, pages 92–101.

B. Wellner, J. Pustejovski, A. Havasi, A. Rumshisky, and R. Suair. 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In *Proc. of SIGDIAL2006*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*.

W. Wright. 2008. *A grammar of the Arabic language*. Bibliobazaar.

Nianwen Xue. 2005. Annotating discourse connectives in the chinese treebank. In *CorpusAnno '05: Proceedings of the Workshop on Frontiers in Corpus Annotations II*, pages 84–91, Morristown, NJ, USA. Association for Computational Linguistics.

D. Zeyrek and B. Webber. 2008. A discourse resource for turkish: Annotating discourse connectives in the metu corpus. *Proceedings of IJCNLP-2008. Hyderabad, India*.

Z. Zhou, Y. Xu, Z. Niu, M. Lan, . Su, and Tan. C. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proc. of Coling 2010*, pages 1507–1514.

Classifying Sentences as Speech Acts in Message Board Posts

Ashequl Qadir and Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
{asheq,riloff}@cs.utah.edu

Abstract

This research studies the text genre of message board forums, which contain a mixture of expository sentences that present factual information and conversational sentences that include communicative acts between the writer and readers. Our goal is to create sentence classifiers that can identify whether a sentence contains a speech act, and can recognize sentences containing four different speech act classes: *Commissives*, *Directives*, *Expressives*, and *Representatives*. We conduct experiments using a wide variety of features, including lexical and syntactic features, speech act word lists from external resources, and domain-specific semantic class features. We evaluate our results on a collection of message board posts in the domain of veterinary medicine.

1 Introduction

In the 1990's, the natural language processing community shifted much of its attention to corpus-based learning techniques. Since then, most of the text corpora that have been annotated and studied are collections of *expository text* (e.g., news articles, scientific literature, etc.). The intent of expository text is to present or explain information to the reader. In recent years, there has been a growing interest in text genres that originate from Web sources, such as weblogs and social media sites (e.g., tweets). These text genres offer new challenges for NLP, such as the need to handle informal and loosely grammatical text, but they also pose new opportunities to study discourse and pragmatic phenomena that are fundamentally different in these genres.

Message boards are common on the WWW as a forum where people ask questions and post comments to members of a community. They are typically devoted to a specific topic or domain, such as finance, genealogy, or Alzheimer's disease. Some message boards offer the opportunity to pose questions to domain experts, while other communities are open to anyone who has an interest in the topic.

From a natural language processing perspective, message board posts are an interesting hybrid text genre because they consist of both expository text and conversational text. Most obviously, the conversations appear as a thread, where different people respond to each other's questions in a sequence of posts. Studying the conversational threads, however, is not the focus of this paper. Our research addresses the issue of conversational pragmatics within individual message board posts.

Most message board posts contain both expository sentences as well as speech acts. The person posting a message (*the "writer"*) often engages in speech acts with the readers. The writer may explicitly greet the readers (*"Hi everyone!"*), request help from the readers (*"Anyone have a suggestion?"*), or commit to a future action (*"I promise I will report back soon."*). But most posts contain factual information as well, such as general knowledge or personal history describing a situation, experience, or predicament.

Our research goals are twofold: (1) to distinguish between expository sentences and speech act sentences in message board posts, and (2) to clas-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 748–758,
Edinburgh, Scotland, UK, July 27–31, 2011.

sify speech act sentences into four types: *Commissives*, *Directives*, *Expressives*, and *Representatives*, following Searle's original taxonomy (Searle, 1976). Speech act classification could be useful for many applications. Information extraction systems could benefit from filtering speech act sentences (e.g., promises and questions) so that facts are only extracted from the expository text. Identifying *Directive* sentences could be used to summarize the questions being asked in a forum over a period of time. *Representative* sentences could be extracted to highlight the conclusions and beliefs of domain experts in response to a question.

In this paper, we present sentence classifiers that can identify speech act sentences and classify them as *Commissive*, *Directive*, *Expressive*, and *Representative*. First, we explain how each speech act class is manifested in message board posts, which can be different from how they occur in spoken dialogue. Second, we train classifiers to identify speech act sentences using a variety of lexical, syntactic, and semantic features. Finally, we evaluate our system on a collection of message board posts in the domain of veterinary medicine.

2 Related Work

There has been relatively little work on applying speech act theory to written text genres, and most of the previous work has focused on email classification. Cohen et al. (2004) introduced the notion of "email speech acts" defined as specific verb-noun pairs following a pre-designed ontology. They approached the problem as a document classification task. Goldstein and Sabin (2006) adopted this notion of email acts (Cohen et al., 2004) but focused on verb lexicons to classify them. Carvalho and Cohen (2005) presented a classification scheme using a dependency network, capturing the sequential correlations with the context emails using transition probabilities from or to a target email. Carvalho and Cohen (2006) later employed N-gram sequence features to determine which N-grams are meaningfully related to different email speech acts with a goal towards improving their earlier email classification based on the writer's intention.

Lampert et al. (2006) performed speech act classification in email messages following a verbal response modes (VRM) speech act taxonomy. They also provided a comparison of VRM taxonomy with Searle's taxonomy (Searle, 1976) of speech act classes. They evaluated several machine learning algorithms using syntactic, morphological, and lexical features. Mildinhall and Noyes (2008) presented a stochastic speech act model based on verbal response modes (VRM) to classify email intentions.

Some research has considered speech act classes in other means of online conversations. Twitchell and Jr. (2004) and Twitchell et al. (2004) employed speech act profiling by plotting potential dialogue categories in a radar graph to classify conversations in instant messages and chat rooms. Nastri et al. (2006) performed an empirical analysis of speech acts in the away messages of instant messenger services to achieve a better understanding of the communication goals of such services. Ravi and Kim (2007) employed speech act profiling in online threaded discussions to determine message roles and to identify threads with questions, answers, and unanswered questions. They designed their own speech act categories based on their analysis of student interactions in discussion threads.

The work most closely related to ours is the research of Jeong et al. (2009) on semi-supervised speech act recognition in both emails and forums. Like our work, their research also classifies individual sentences, as opposed to entire documents. However, they trained their classifier on spoken telephone (SWBD-DAMSL corpus) and meeting (MRDA corpus) conversations and mapped the labelled dialog act classes of these corpora to 12 dialog act classes that they found suitable for email and forum text genres. These dialog act classes (addressed as speech acts by them) are somewhat different from Searle's original speech act classes. They also used substantially different types of features than we do, focusing primarily on syntactic subtree structures.

3 Classifying Speech Acts in Message Board Posts

3.1 Speech Act Class Definitions

Searle's (Searle, 1976) early research on *speech acts* was seminal work in natural language processing that opened up a new way of thinking about con-

versational dialogue and communication. Our goal was to try and use Searle's original speech act definitions and categories as the basis for our work to the greatest extent possible, allowing for some interpretation as warranted by the WWW message board text genre.

For the purposes of defining and evaluating our work, we created detailed annotation guidelines for four of Searle's speech act classes that commonly occur in message board posts: *Commissives*, *Directives*, *Expressives*, and *Representatives*. We omitted the fifth of Searle's original speech act classes, *Declarations*, because we virtually never saw declarative speech acts in our data set.[1] The data set used in our study is a collection of message board posts in the domain of veterinary medicine. We designed our definitions and guidelines to reflect language use in the text genre of message board posts, trying to be as domain-independent as possible so that these definitions should also apply to message board texts representing other topics. However, we give examples from the veterinary domain to illustrate how these speech act classes are manifested in our data set.

Commissives: A *Commissive speech act* occurs when the speaker commits to a future course of action. In conversation, common Commissive speech acts are promises and threats. In message boards, these types of Commissives are relatively rare. However, we found many statements where the main purpose was to confirm to the readers that the writer would perform some action in the future. For example, a doctor may write *"I plan to do surgery on this patient tomorrow"* or *"I will post the test results when I get them later today"*. We viewed such statements as implicit commitments to the reader about intended actions. We also considered decisions not to take an action as Commissive speech acts (e.g., *"I will not do surgery on this cat because it would be too risky."*). However, statements indicating that an action will not occur because of circumstances beyond the writer's control were considered to be factual statements and not speech acts (e.g., *"I cannot do an ultrasound because my machine is broken."*).

Directives: A *Directive speech act* occurs when the speaker expects the listener to do something as a response. For example, the speaker may ask a question, make a request, or issue an invitation. Directive speech acts are common in message board posts, especially in the initial post of each thread when the writer explicitly requests help or advice regarding a specific topic. Many Directive sentences are posed as questions, so they are easy to identify by the presence of a question mark. However, the language in message board forums is informal and often ungrammatical, so many Directives are posed as a question but do not end in a question mark (e.g., *"What do you think."*). Furthermore, many Directive speech acts are not stated as a question but as a request for assistance. For example, a doctor may write *"I need your opinion on what drug to give this patient."* Finally, some sentences that end in question marks are rhetorical in nature and do not represent a Directive speech act, such as *"Can you believe that?"*.

Expressives: An *Expressive speech act* occurs in conversation when a speaker expresses his or her psychological state to the listener. Typical cases are when the speaker thanks, apologizes, or welcomes the listener. Expressive speech acts are common in message boards because writers often greet readers at the beginning of a post (*"Hi everyone!"*) or express gratitude for help from the readers (*"I really appreciate the suggestions."*). We also found Expressive speech acts in a variety of other contexts, such as apologies.

Representatives: According to Searle, a *Representative speech act* commits the speaker to the truth of an expressed proposition. It represents the speaker's belief of something that can be evaluated to be true or false. These types of speech acts were less common in our data set, but some cases did exist. In the veterinary domain, we considered sentences to be a Representative speech act when a doctor explicitly confirmed a diagnosis or expressed their suspicion or hypothesis about the presence (or absence) of a disease or symptom. For example, if a doctor writes that *"I suspect the patient has pancreatitis."* then this represents the doctor's own proposition/belief about what the disease might be.

Many sentences in our data set are stated as fact but could be reasonably inferred to be speech acts. For example, suppose a doctor writes *"The cat has*

[1] Searle defines Declarative speech acts as statements that bring about a change in status or condition to an object by virtue of the statement itself. For example, a statement declaring war or a statement that someone is fired.

pancreatitis.". It would be reasonable to infer that the doctor writing the post diagnosed the cat with pancreatitis. And in many cases, that is true. However, we saw many posts where that inference would have been wrong. For example, the following sentence might say *"The cat was diagnosed by a previous vet but brought to me due to new complications"* or *"The cat was diagnosed with it 8 years ago as a kitten in the animal shelter"*. Consequently, we were very conservative in labelling sentences as Representative speech acts. Any sentence presented as fact was not considered to be a speech act. A sentence was only labelled as a Representative speech act if the writer explicitly expressed his belief.

3.2 Features for Speech Act Classification

To create speech act classifiers, we designed a variety of lexical, syntactic, and semantic features. We tried to capture linguistic properties associated with speech act expressions as well as discourse properties associated with individual sentences and the message board post as a whole. We also incorporated speech act word lists that were acquired from external resources, and used two types of semantic features to represent semantic entities associated with the veterinary domain. Except for the semantic features, all of our features are domain-independent so should be able to recognize speech act sentences across different domains. We experimented with domain-specific semantic features to test our hypothesis that Commissive speech acts can be associated with domain-specific semantic entities.

For the purposes of analysis, we partition the feature set into three groups: *Lexical and Syntactic (LexSyn) Features*, *Speech Act Clue Features*, and *Semantic Features*. Unless otherwise noted, all of the features had binary values indicating the presence or absence of that feature.

3.2.1 Lexical and Syntactic Features

We designed a variety of features to capture lexical and syntactic properties of words and sentences. We described the feature set below, with the features categorized based on the type of information that they capture.

Unigrams: We created bag-of-word features representing each unigram in the training set. Numbers were replaced with a special # token.

Personal Pronouns: We defined three features to look for the presence of a 1st person pronoun, 2nd person pronoun, and 3rd person pronoun. We included the subjective, objective, and possessive form of each pronoun (e.g., *he*, *him*, and *his*).

Tense: Speech acts such as Commissives can be related to tense. We created three features to identify verb phrases that occur in the *past*, *present*, or *future* tense. To recognize tense, we followed the rules defined by Allen (1995).

Tense + Person: We created four features that require the presence of a first person subjective pronoun (I, we) within a two word window on the left of a verb phrase matching one of four tense representations: *past*, *present*, *future*, and *present progressive* (a subset of the more general *present* tense representation).

Modals: One feature indicates whether the sentence contains a modal (*may*, *must*, *shall*, *will*, *might*, *should*, *would*, *could*).

Infinitive VP: One feature looks for an infinitive verb phrase ('to' followed by a verb) that is preceded by a first person pronoun (I, we) within a three word window on the left. This feature tries to capture common Commissive expressions (e.g., *"I definitely plan to do the test tomorrow."*).

Plan Phrases: Commissives are often expressed as a plan, so we created a feature that recognizes four types of plan expressions: *"I am going to"*, *"I am planning to"*, *"I plan to"*, and *"My plan is to"*.

Sentence contains Early Punctuation: One feature checks for the following punctuation marks within the first three tokens of the sentence: , : ! This feature was designed to recognize greetings, such as: *"Hi,"* , or *"Hiya everyone !"*.

Sentence begins with Modal/Verb: One feature checks if a sentence begins with a modal or verb. The intuition is to capture interrogative and imperative sentences, since they are likely to be Directives.

Sentence begins with WH Question: One feature checks if a sentence begins with a WH question word (Who, When, Where, What, Which, What, How).

Neighboring Question: One feature checks whether the following sentence contains a question mark '?'. We observed that in message boards, *Directives* often occur in clusters.

Sentence Position: Four binary features represent the relative position of the sentence in the post. One feature indicates whether it is the first sentence, one feature indicates whether it is the last sentence, one feature indicates whether it is the second to last sentence, and one feature indicates whether the sentence occurs in the bottom 25% of the message. The motivation for these features is that Expressives often occur at the beginning and end of the post, and Directives tend to occur toward the end.

Number of Verbs: One feature represents the number of verbs in the sentence using four possible values: 0, 1, 2, >2. Some speech acts classes (e.g., Expressives) may occur with no verbs, and rarely occur in long, complex sentences.

3.2.2 Speech Act Word Clues

We collected speech act word lists (mostly verbs) from two external sources. In Searle's original paper (Searle, 1976), he listed words that he considered to be indicative of speech acts. We discarded a few that we considered to be overly general, and we added a few additional words. We also collected a list of speech act verbs published in (Wierzbicka, 1987). The details for these *speech act clue lists* are given below. Our system recognized all derivations of these words.

Searle Keywords: We created one feature for each speech act class. The Representative keywords were: (*hypothesize*, *insist*, *boast*, *complain*, *conclude*, *deduce*, *diagnose*, and *claim*). We discarded 3 words from Searle's list (*suggest*, *call*, *believe*) and added 2 new words, *assume* and *suspect*. The Directive keywords were: (*ask*, *order*, *command*, *request*, *beg*, *plead*, *pray*, *entreat*, *invite*, *permit*, *advise*, *dare*, *defy*, *challenge*). We added the word *please*. The Expressives keywords were: (*thank*, *apologize*, *congratulate*, *condole*, *deplore*, *welcome*). We added the words *appreciate* and *sorry*. Searle did not provide any hint on possible indicator words for Commissives, so we manually defined five likely Commissive keywords: (*plan*, *commit*, *promise*, *tomorrow*, *later*).

Wierzbicka Verbs: We created one feature that included 228 speech act verbs listed in the book *"English speech act verbs: a semantic dictionary"* (Wierzbicka, 1987)[2].

3.2.3 Semantic Features

All of the previous features are domain-independent and should be useful for identifying speech acts sentences across many domains. However, we hypothesized that semantic entities may correlate with speech acts within a particular domain. For example, consider medical domains. Representative speech acts may involve diagnoses and hypotheses regarding diseases and symptoms. Similarly, Commissive speech acts may reveal a doctor's plan or intention regarding the administration of drugs or tests. Thus, it may be beneficial for a classifier to know whether a sentence contains certain semantic entities. We experimented with two different sources of semantic information.

Semantic Lexicon: Basilisk (Thelen and Riloff, 2002) is a bootstrapping algorithm that has been used to induce semantic lexicons for terrorist events (Thelen and Riloff, 2002), biomedical concepts (McIntosh, 2010), and subjective/objective nouns for opinion analysis (Riloff et al., 2003). We ran Basilisk over our collection of 15,383 veterinary message board posts to create a semantic lexicon for veterinary medicine. As input, Basilisk requires seed words for each semantic category. To obtain seeds, we parsed the corpus using a noun phrase chunker, sorted the head nouns by frequency, and manually identified the 20 most frequent nouns belonging to four semantic categories: DISEASE/SYMPTOM, DRUG, TEST, and TREATMENT.

However, the induced TREATMENT lexicon was of relatively poor quality so we did not use it. The DISEASE/SYMPTOM lexicon appeared to be of good quality, but it did not improve the performance of our speech act classifiers. We suspect that this is due to the fact that diseases were not distinguised from symptoms in our lexicon.[3] Representative speech acts are typically associated with disease diagnoses

[2] `openlibrary.org/b/OL2413134M/English_speech_act_verbs`

[3] We induced a single lexicon for diseases and symptoms because it is difficult to draw a clear line between them semantically. A veterinary consultant explained to us that the same term (e.g., diabetes) may be considered a symptom in one context if it is secondary to another condition (e.g., pancreatitis) but a disease in a different context if it is the primary diagnosis.

and hypotheses, rather than individual symptoms.

In the end, we only used the DRUG and TEST semantic lexicon in our classifiers. We used all 1000 terms in the DRUG lexicon, but only used the top 200 TEST words because the quality of the lexicon seemed questionable after that point.

Semantic Tags: We also used bootstrapped contextual semantic taggers (Huang and Riloff, 2010) that had been previously trained for the domain of veterinary medicine. These taggers assign semantic class labels to noun phrase instances based on the surrounding context in a sentence. The taggers were trained on 4,629 veterinary message board posts using 10 seed words for each semantic category (see (Huang and Riloff, 2010) for details). To ensure good precision, only tags that have a confidence value ≥ 1.0 were used. Our speech act classifiers used the tags associated with two semantic categories: DRUG and TEST.

3.3 Classification

To create our classifiers, we used the Weka (Hall et al., 2009) machine learning toolkit. We used Support Vector Machines (SVMs) with a polynomial kernel and the default settings supplied by Weka. Because a sentence can include multiple speech acts, we created a set of binary classifiers, one for each of the four speech act classes. All four classifiers were applied to each sentence, so a sentence could be assigned multiple speech act classes.

4 Evaluation

4.1 Data Set

Our data set consists of message board posts from the Veterinary Information Network (VIN), which is a web site (www.vin.com) for professionals in veterinary medicine. Among other things, VIN hosts message board forums where veterinarians and other veterinary professionals can discuss issues and pose questions to each other. Over half of the small animal veterinarians in the U.S. and Canada use the VIN web site.

We obtained 15,383 VIN message board threads representing three topics: cardiology, endocrinology, and feline internal medicine. We did basic cleaning, removing html tags and tokenizing numbers. We then applied the Stanford part-of-speech tagger (Toutanova et al., 2003) to each sentence to obtain part-of-speech tags for the words. For our experiments, we randomly selected 150 message board threads from this collection. Since the goal of our work was to study speech acts in sentences, and not the conversational dialogue between different writers, we used only the initial post of each thread. These 150 message board posts contained a total of 1,956 sentences, with an average of 13.04 sentences per post. In the next section, we explain how we manually annotated each sentence in our data set to create gold standard speech act labels.

4.2 Gold Standard Annotations

To create training and evaluation data for our research, we asked two human annotators to manually label sentences in our message board posts. Identifying speech acts is not always obvious, even to people, so we gave them detailed annotation guidelines describing the four speech act classes discussed in Section 3.1. Then we gave them the same set of 50 message board posts from our collection to annotate independently. Each annotator was told to assign one or more speech act classes to each sentence (COM, DIR, EXP, REP), or to label the sentence as having no speech acts (NONE). The vast majority of sentences had either no speech acts or at most one speech act, but a small number of sentences contained multiple types of speech acts.

We measured the inter-annotator agreement of the two human judges using the kappa (κ) score (Carletta, 1996). However, kappa agreement scores are only applicable to labelling schemes where each instance receives a single label. Therefore we computed kappa agreement in two different ways to look at the results from two different perspectives. In the first scheme, we discarded the small number of sentences that had multiple speech act labels and computed kappa on the rest.[4] This produced a kappa score of .95, suggesting extremely high agreement. However, over 70% of the sentences in our data set have no speech act at all, so NONE was by far the most common label. Consequently, this agreement score does not necessarily reflect how consistently the judges agreed on the four speech act classes.

[4]Of the 594 sentences in these 50 posts, only 22 sentences contained multiple speech act classes.

In the second scheme, we computed kappa for each speech act category independently. For each category C, the judges were considered to be in agreement if both of them assigned category C to the sentence or if neither of the judges assigned category C to the sentence. Table 1 shows the κ agreement scores using this approach.

Speech Act	Kappa (κ) score
Expressive	.97
Directive	.94
Commissive	.81
Representative	.77

Table 1: Inter-annotator (κ) agreement

Inter-annotator agreement was very high for both the Expressive and Directive classes. Agreement was lower for the Commissive and Representative classes, but still relatively good so we felt comfortable that we had high-quality annotations.

To create our final data set, the two judges adjudicated their disagreements on this set of 50 posts. We then asked each annotator to label an additional (different) set of 50 posts each. All together, this gave us a gold standard data set consisting of 150 annotated message board posts. Table 2 shows the distribution of speech act labels in our data set. 71% of the sentences did not include any speech acts. These were usually expository sentences containing factual information. 29% of the sentences included one or more speech acts, so nearly $\frac{1}{3}$ of the sentences were conversational in nature. Directive and Expressive speech acts are by far the most common, with nearly 26% of all sentences containing one of these speech acts. Commissive and Representative speech acts are less common, each occurring in less than 3% of the sentences.[5]

4.3 Experimental Results

4.3.1 Speech Act Filtering

For our first experiment, we created a *speech act filtering classifier* to distinguish sentences that contain one or more speech acts from sentences that do not contain any speech acts. Sentences labelled as having one or more speech acts were positive instances, and sentences labelled as NONE were negative instances. Speech act filtering could be useful for many applications, such as information extraction systems that only seek to extract facts. For example, information may be posed as a question (in a Directive) rather than a fact, information may be mentioned as part of a future plan (in a Commissive) that has not actually happened yet, or information may be stated as a hypothesis or suspicion (in a Representative) rather than as a fact.

Speech Act	# sentences	distribution
None	1397	71.42%
Directive	311	15.90%
Expressive	194	9.92%
Representative	57	2.91%
Commissive	51	2.61%

Table 2: Speech act class distribution in our data set.

We performed 10-fold cross validation on our set of 150 annotated message board posts. Initially, we used all of the features defined in Section 3.2. However, during the course of our research we discovered that only a small subset of the lexical and syntactic features seemed to be useful, and that removing the unnecessary features improved performance. So we created a subset of *minimal lexsyn features*, which will be described in Section 4.3.2. For speech act filtering, we used the *minimal lexsyn features* plus the speech act clues and semantic features.[6]

Class	P	R	F
Speech Act	.86	.83	.84
No Speech Act	.93	.95	.94

Table 3: Precision, Recall, F-measure for speech act filtering.

Table 3 shows the performance for speech act filtering with respect to Precision (P), Recall (R), and F-measure score (F).[7] The classifier performed well, recognizing 83% of the speech act sentences with 86% precision, and 95% of the expository (no

[5]These numbers do not add up to 100% because some sentences contain multiple speech acts.

[6]This is the same feature set used to produce the results for row E of Table 4.

[7]We computed an F1 score with equal weighting of precision and recall.

	Features	Commissives			Directives			Expressives			Representatives		
		P	R	F	P	R	F	P	R	F	P	R	F
	Baselines												
	Com baseline	.45	.08	.14	-	-	-	-	-	-	-	-	-
	Dir baseline	-	-	-	.97	.73	.83	-	-	-	-	-	-
	Exp baseline 1	-	-	-	-	-	-	.58	.18	.28	-	-	-
	Exp baseline 2	-	-	-	-	-	-	.97	.86	.91	-	-	-
	Rep baseline	-	-	-	-	-	-	-	-	-	1.0	.05	.10
	Classifiers												
U	Unigram	.45	.20	.27	.87	.84	.85	.97	.88	.92	.32	.12	.18
A	U+all lexsyn	.52	.33	.40	.87	.84	**.86**	.98	.88	.92	.30	.14	.19
B	U+minimal lexsyn	.59	.33	.42	.87	.85	**.86**	.98	.88	.92	.32	.14	.20
C	B+speechActClues	.57	.31	.41	.86	.84	.85	.97	.91	**.94**	.33	.16	**.21**
D	C+semTest	.64	.35	.46	.87	.84	.85	.97	.91	**.94**	.33	.16	**.21**
E	D+semDrug	.63	.39	**.48**	.86	.84	.85	.97	.91	**.94**	.32	.16	**.21**

Table 4: Precision, Recall, F-measure for four speech act classes. The highest F score for each category appears in boldface.

speech act) sentences with 93% precision.

4.3.2 Speech Act Categorization

BASELINES

Our next set of experiments focused on labelling sentences with the four specific speech act classes: *Commissive*, *Directive*. *Expressive*, and *Representative*. To assess the difficulty of identifying each speech act category, we created several simple baselines using our intuitions about each category.

For Commissives, we created a heuristic to capture the most obvious cases of future tense (because Commissive speech acts represent a writer's commitment toward a future course of action). For example, the presence of the phrases 'I will' and 'I shall' were hypothesized by Cohen et al. (2004) to be useful bigram clues for Commissives. This baseline looks for future tense verb phrases with a 1st person pronoun within one or two words preceding the verb phrase. The **Com** baseline row of Table 4 shows the results for this heuristic, which obtained 8% recall with 45% precision. The heuristic applied to only 9 sentences in our test set, 4 of which contained a Commissive speech act.

Directive speech acts are often questions, so we created a baseline system that labels all sentences containing a question mark as a Directive. The **Dir** baseline row of Table 4 shows that 97% of sentences with a question mark were indeed Directives.[8] But only 73% of the Directive sentences contained a question mark. The remaining 27% of Directives did not contain a question mark and generally fell into two categories. Some sentences asked a question but the writer ended the sentence with a period (e.g., *"Has anyone seen this before."*). And many directives were expressed as requests rather than questions (e.g., *"Let me know if anyone has a suggestion."*).

For Expressives, we implemented two baselines. **Exp** baseline 1 simply looks for an exclamation mark, but this heuristic did not work well (18% recall with 58% precision) because exclamation marks were often used for general emphasis (e.g., *"The owner is frustrated with cleaning up urine!"*). **Exp** baseline 2 looks for the presence of four common expressive words (*appreciate*, *hi*, *hello*, *thank*), including morphological variations of *appreciate* and *thank*. This baseline produced very good results, 86% recall with 97% precision. Obviously a small set of common expressions account for most of the Expressive speech acts in our corpus. However, the word "hi" did produce some false hits because it was used as a shorthand for "high", usually when reporting test results (e.g., *"hi calcium"*).

[8] 235 sentences contained a question mark, and 227 of them were Directives.

Finally, as a baseline for the Representative class we simply looked for the words *diagnose(d)* and *suspect(ed)*. The **Rep** baseline row of Table 4 shows that this heuristic was 100% accurate, but only produced 5% recall (matching 3 of the 57 Representative sentences in our test set).

CLASSIFIER RESULTS

The bottom portion of Table 4 shows the results for our classifiers. As we explained in Section 3.3, we created one classifier for each speech act category, and all four classifiers were applied to each sentence. So a sentence could receive anywhere from 0-4 speech act labels indicating how many different types of speech acts appeared in the sentence. We trained and evaluated each classifier using 10-fold cross-validation on our gold standard data set.

The *Unigram (U)* row shows the performance of classifiers that use only unigram features. For Directives, we see a 2% F-score improvement over the baseline, which reflects a recall gain of 11% but a corresponding precision loss of 10%. The unigrams are clearly helpful in identifying many Directive sentences that do not end in a question mark, but at some cost to accuracy. For Expressives, the unigram classifier achieves an F score of 92%, identifying slightly more Expressive sentences than the baseline with the same level of precision. For Commissives and Representatives, the unigram classifiers performed susbtantially better than their corresponding baseline systems, but performance is still relatively weak.

Row A (*U+ all lexsyn*) in Table 4 shows the results using unigram features plus all of the lexical and syntactic features described in Section 3.2.1. The lexical and syntactic features dramatically improve performance on Commissives, increasing F score from 27% to 40%, and they produce a 2% recall gain for Representatives but with a corresponding loss of precision.

However, we observed that only a few of the lexical and syntactic features had much impact on performance. We experimented with different subsets of the features and obtained even better performance when using just 10 of them, which we will refer to as the *minimal lexsyn* features. The *minimal lexsyn* feature set consists of the 4 Tense+Person features, the Early Punctuation feature, the Sentence begins with Modal/Verb feature, and the 4 Sentence Position features. Row B shows the results using unigram features plus only these *minimal lexsyn* features. Precision improves for Commissives by an additional 7% and Representatives by 2% when using only these lexical and syntactic features. Consequently, we use the *minimal lexsyn* features for the rest of our experiments.

Row C shows the results of adding the speech act clue words (see Section 3.2.2) to the feature set used in Row B. The speech act clue words produced an additional recall gain of 3% for Expressives and 2% for Representatives, although performance on Commissives dropped 2% in both recall and precision.

Rows D and E show the results of adding the semantic features. We added one semantic category at a time to measure the impact of them separately. Row D adds two semantic features for the TEST category, one from the Basilisk lexicon and one from the semantic tagger. The TEST semantic features produced an F-score gain of 5% for Commissives, improving recall by 4% and precision by 7%. Row E adds two semantic features for the DRUG category. The DRUG features produced an additional F-score gain of 2% for Commissives, improving recall by 4% with a slight drop in precision.

4.4 Analysis

Together, the TEST and DRUG semantic features dramatically improved the classifier's ability to recognize Commissive speech acts, increasing its F score from 41% $\rightarrow$ 48%. This result demonstrates that in the domain of veterinary medicine, some types of semantic entities are associated with speech acts. Our intuition behind this result is that commitments are usually related to future actions. In veterinary medicine, TESTS and DRUGS are associated with actions performed by doctors. Doctors help their patients by prescribing or administering drugs and by conducting tests. So these semantic entities may serve as a proxy to implicitly represent actions that the doctor has done or may do. In future work, explicitly recognizing actions and events many be a worthwhile avenue to further improve results.

We achieved good success at identifying both Directives and Expressives, although simple heuristics also perform well on these categories. We showed that training a Directive classifier can help to iden-

tify Directive sentences that do not end with a question mark, although at the cost of some precision.

The Commissive speech act class benefitted the most from the rich feature set. Unigrams are clearly not sufficient to identify Commissive sentences. Many different types of clues seem to be important for recognizing these sentences. The improvements obtained from adding semantic features also suggests that domain-specific semantics can be useful for recognizing some speech acts. However, there is still ample room for improvement, illustrating that speech act classification is a challenging problem.

Representative speech acts were by far the most difficult to recognize. We believe that there are several reasons for their low performance. First, Representatives were sparse in the data set, occurring in only 2.91% of the sentences. Consequently, the classifier had relatively few positive training instances. Second, Representatives had the lowest inter-annotator agreement, indicating that human judges had difficulty recognizing these speech acts too. The judges often disagreed about whether a hypothesis or suspicion was the writer's own belief or whether it was stated as a fact reflecting general medical knowledge. The message board text genre is especially challenging in this regard because the writer is often presumed to be expressing his/her beliefs even when the writer does not explicitly say so. Finally, our semantic features could not distinguish between diseases and symptoms. Access to a resource that can reliably identify disease terms could potentially improve performance in this domain.

5 Conclusions

Our goal was to identify speech act sentences in message board posts and to classify the sentences with respect to four categories in Searle's (1976) speech act taxonomy. We achieved good results for speech act filtering and the identification of Directive and Expressive speech act sentences. We found that Representative and Commissive speech acts are much more difficult to identify, although the performance of our Commissive classifier substantially improved with the addition of lexical, syntactic, and semantic features. Except for the semantic class information, our feature set is domain-independent and could be used to recognize speech act sentences in message boards for any domain. Furthermore, our features only rely on part-of-speech tags and do not require parsing, which is of practical importance for text genres such as message boards that are littered with ungrammatical text, typos, and shorthand notations.

In future work, we believe that segmenting sentences into clauses may help to train classifiers more precisely. Ultimately, we would like to identify the speech act expressions themselves because some sentences contain speech acts as well as factual information. Extracting the speech act expressions and clauses from message boards and similar text genres could provide better tracking of questions and answers in web forums and be used for summarization.

6 Acknowledgments

We gratefully acknowledge that this research was supported in part by the National Science Foundation under grant IIS-1018314. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the U.S. government.

References

James Allen. 1995. *Natural language understanding (2nd ed.)*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Comput. Linguist.*, 22:249–254, June.

Vitor R. Carvalho and William W. Cohen. 2005. On the collective classification of email "speech acts". In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 345–352, New York, NY, USA. ACM Press.

Vitor R. Carvalho and William W. Cohen. 2006. Improving "email speech acts" analysis via n-gram selection. In *Proceedings of the HLT-NAACL 2006 Workshop on Analyzing Conversations in Text and Speech*, ACTS '09, pages 35–41, Stroudsburg, PA, USA. Association for Computational Linguistics.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "speech acts". In *EMNLP*, pages 309–316. ACL.

Jade Goldstein and Roberta Evans Sabin. 2006. Using speech acts to categorize email and identify email gen-

res. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 03*, pages 50.2–, Washington, DC, USA. IEEE Computer Society.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November.

Ruihong Huang and Ellen Riloff. 2010. Inducing domain-specific semantic class taggers from (almost) nothing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 275–285, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Semi-supervised speech act recognition in emails and forums. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1250–1259, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andrew Lampert, Robert Dale, and Cecile Paris. 2006. Classifying speech acts using verbal response modes. In *Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006)*, pages 34–41. Sydney Australia : ALTA.

Tara McIntosh. 2010. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 356–365, Stroudsburg, PA, USA. Association for Computational Linguistics.

John Mildinhall and Jan Noyes. 2008. Toward a stochastic speech act model of email behavior. In *CEAS*.

Jacqueline Nastri, Jorge Pena, and Jeffrey T. Hancock. 2006. The construction of away messages: A speech act analysis. *J. Computer-Mediated Communication*, pages 1025–1045.

Sujith Ravi and Jihie Kim. 2007. Profiling student interactions in threaded discussions with speech act classifiers. In *Proceeding of the 2007 conference on Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work*, pages 357–364, Amsterdam, The Netherlands, The Netherlands. IOS Press.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

John R. Searle. 1976. A classification of illocutionary acts. *Language in Society*, 5(1):pp. 1–23.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 214–221, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Douglas P. Twitchell and Jay F. Nunamaker Jr. 2004. Speech act profiling: a probabilistic method for analyzing persistent conversations and their participants. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 1–10, January.

Douglas P. Twitchell, Mark Adkins, Jay F. Nunamaker Jr., and Judee K. Burgoon. 2004. Using speech act theory to model conversations for automated classification and retrieval. In *Proceedings of the International Working Conference Language Action Perspective Communication Modelling (LAP 2004)*, pages 121–130.

A. Wierzbicka. 1987. *English speech act verbs: a semantic dictionary*. Academic Press, Sydney, Orlando.

Learning Local Content Shift Detectors from Document-level Information

Richárd Farkas
Institute for Natural Language Processing
University of Stuttgart
farkas@ims.uni-stuttgart.de

Abstract

Information-oriented document labeling is a special document multi-labeling task where the target labels refer to a specific information instead of the topic of the whole document. These kind of tasks are usually solved by looking up indicator phrases and analyzing their local context to filter false positive matches. Here, we introduce an approach for machine learning *local content shifters* which detects irrelevant local contexts using just the original document-level training labels. We handle content shifters in general, instead of learning a particular language phenomenon detector (e.g. negation or hedging) and form a single system for document labeling and content shift detection. Our empirical results achieved 24% error reduction – compared to supervised baseline methods – on three document labeling tasks.

1 Introduction

There are special document multi-labeling tasks where the target labels refer to a specific piece of information extractable from the document instead of the overall topic of the document. In these kinds of tasks the target information is usually an attribute or relation related to the target entity (usually a person or an organisation) of the document in question, but the task is to assign class labels at the document (entity) level. For example, the smoking habits of the patients are frequently discussed in the textual parts of clinical notes (Uzuner et al., 2008). In this case the task is to find specific information in the text – i.e. the patient in question is a smoker, past smoker, non-smoker – but at the end an application has to assign labels to the documents(patients). Similarly, the soccer club names where a sportsman played for are document(sportman)-level labels in Wikipedia articles expressed by the Wikipedia categories. The target information in these tasks is usually just mentioned in the document and much of the document is irrelevant for this information request in contrast to standard document classification tasks where the goal is to identify the topics of the whole document. On the other hand, they are not a standard information extraction task as the task is to assign class labels to documents, and the training dataset contains labels just at this level. These special tasks lie somewhere between information extraction and document classification and require special approaches to solve them. We will call them *Information-oriented document labeling* throughout this paper. There are several application areas where information-oriented document labels are naturally present in an enormous amount like clinical records, Wikipedia categories and user-generated tags of news.

Previous evaluation campaigns (Uzuner et al., 2008; Pestian et al., 2007; Uzuner, 2009) demonstrated that information-oriented document labeling can be effectively performed by looking up *indicator phrases* which can be gathered by hand, by corpus statistics or in a hybrid way. However these campaigns also highlighted that the analysis of the *local context* of the indicator phrases is crucial. For instance, in the smoking habit detection task there are a few indicator words (e.g. *smokes*, *cigarette*) and the local context of their occurrences in texts should

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 759–770,
Edinburgh, Scotland, UK, July 27–31, 2011.

be analysed to see whether their semantic was radically changed (e.g. they are negated or in a past tense), for instance:

> *The patient has a 20 pack-year smoking history.*
>
> *The patient denies any smoking history.*
>
> *He has a greater than 100 pack year smoking history and quit 9 to 10 years ago.*

We propose a simple but efficient approach for information-oriented document labeling tasks by addressing the automatic detection of language phenomena for a particular task which alters the sense or information content of the indicator phrase's occurrences. For example, they may be logical modifiers (e.g. negation) or modal modifiers (e.g. auxiliaries like *might* and *can*); they may refer to a subject which differs from the target entity of the task (e.g. clinical notes usually contain information about the family history of the patient); or the semantic content of the shifter may change the role of the target span of a text (e.g. a sportsman can play *for* or *against* a particular team). We call these phenomena *content shifters* and the task of identifying them *content shift detection (CSD)*.

Existing CSD approaches focus on a particular class of language phenomena (especially negation or hedging) and use hand-crafted rules (Chapman et al., 2007) or a supervised learning approach that exploits corpora manually annotated at the token-level for a particular type of content shifter (Morante et al., 2009). Moreover higher level applications (like document labeling and information extraction) use a separate CSD module which is developed independently from the target task. We argue that the nature of content shifters is domain and task dependent, so training corpora (at the token-level) are required for content shifters which are important for a particular task but the construction of such training corpora is expensive. Here, we propose an alternative approach which uses only document-level labels.

The input of our system is a training corpus labeled on the document level (e.g. a clinical dataset consisting clinical notes and meta-data about patients). Our approach extracts indicator phrases and trains a CSD jointly. We focus on local content shifters and we analyse just the sentences of indicator phrase occurrences. Our chief assumption is that CSD can be learnt by exploiting the false positive occurrences of indicator phrases in the training dataset. We show that our method performs significantly better than standard document classifiers (which were designed for a slightly different task).

The chief contributions of our work are that (i) we handle the CSD problem in general, so we detect all content shifters instead of focusing on one particular language phenomenon, (ii) we form a single framework for joint CSD and document labeling, (iii) moreover our approach does not require a dedicated annotated training dataset for content shifters.

2 Related Work

Information-oriented document classification tasks were first highlighted in the clinical domain where medical reports contain useful information about the patient in question, but labels are only available at the document (patient) level. The field of clinical NLP has been studied extensively since the 1990s (Larkey and Croft, 1995), but the most recent results are related to the shared task challenges organized relatively recently (Pestian et al., 2007; Uzuner et al., 2008; Uzuner, 2009). For example the first I2B2 challenge in 2006 (Uzuner et al., 2008) focused on the smoking habits of the patient, the CMC challenge in 2007 (Pestian et al., 2007) dealt with the problem of automatically constructing ICD coding systems and the second I2B2 challenge (Uzuner, 2009) addressed the classification of discharge summaries according to the question "Who's obese and what co-morbidities do they have?". These challenges were dominated by entirely or partly rule-based systems that solved the tasks using indicator phrase lookup and incorporated explicit mechanisms for detecting speculation and negation.

Another domain for information-oriented document classification might be Wikipedia, which contains rich information about entities like persons, places or organisations. Some items of information are available about these entities in the form of categories and infoboxes assigned to articles. Automatic document labeling methods can be trained based on these assignments (Schönhofen, 2006), but these labels do not refer to the main theme of the article but

to a certain type of information.

Existing content shift detection approaches focus on a particular class of language phenomena, especially on negation and hedge recognitions. Available tools work mainly on clinical and biological domains. The first systems were fully hand-crafted (Light et al., 2004; Friedman et al., 1994; Chapman et al., 2007) without any empirical evaluation on a dedicated corpus. Recently, there have been several corpora published with manual sentence-, event- or token-level annotation for negation, certainty and factuality in the biological (Medlock and Briscoe, 2007; Vincze et al., 2008), newswire (Strassel et al., 2008; Sauri and Pustejovsky, 2009) and encyclopedical (Farkas et al., 2010) domains.

Exploiting these corpora, machine learning models were also developed. Solving the sentence-level task, Medlock and Briscoe (2007) used single words as input features in order to classify sentences from biological articles as speculative or non-speculative. Szarvas (2008) extended their methodology to use n-gram features and a semi-supervised selection of the keyword features. Ganter and Strube (2009) proposed an approach for the automatic detection of sentences containing uncertainty based on Wikipedia weasel tags and syntactic patterns. For in-sentence negation and speculation detection, Morante et al. (2009) developed scope – i.e. content shifted text spans – detectors for negation and speculation following a supervised sequence labeling approach, while Özgür and Radev (2009) developed a rule-based system that exploits syntactic patterns. The goal of the CoNLL 2010 Shared Task (Farkas et al., 2010) was to develop linguistic scope detectors as well. The participants usually followed a supervised sequence labeling approach or used a rule-based system that exploits syntactic patterns. The approach of classifying identified events into whether they fall under negation or speculation was followed by Sauri and Pustejovsky (2009) and the participants of the BioNLP'09 Shared Task (Kim et al., 2009). Here the systems investigated the syntax path between the event trigger and a cue word (which came from a small lexicon) (Kilicoglu and Bergler, 2009; Aramaki et al., 2009).

Our approach differs from the previous works fundamentally. We deal with the two tasks (information-oriented document classification and content shift detection) together and introduce a co-learning approach for them. Our approach handles content shifters in a data-driven and generalized way i.e. it is not specialized for a certain class of language phenomena. Instead it tries to recognize task-specific syntactic and semantic patterns which are responsible for semantic changes or irrelevance. In addition, we have no access to a gold-standard sentence-level or in-sentence-level annotation but exploit document-level ones.

3 Tasks and Datasets

Before introducing our approach in detail we describe three tasks and datasets which were used in our experiments in order to give an insight into the challenges of the information-oriented document labeling tasks. Table 1 summarizes the key statistical figures (the number of documents in the corpora, the size of the label sets along with the average number of tokens and label assignments per document) of the datasets used for the experimental evaluations.

Table 1: The datasets used in our experiments.

	CMC	Obes	Soccer
domain	clinical	clinical	encycl.
|train|	978	730	4850
|eval|	976	507	1736
#token/d	25	1387	389
#labels	45	16	12
#label/d	1.24	4.37	1.23

The CMC ICD Coding Dataset was originally prepared for a shared task challenge organized by the Computational Medicine Center (CMC) in Cincinatti, Ohio in 2007 (Pestian et al., 2007). It contains radiology reports along with document-level International Classification of Diseases (ICD) codes given by three human experts. ICD is a coding of diseases, signs, symptoms and abnormal findings. In our experiments we used the train/evaluation split of the shared task. The ICD coding guide states that negative or uncertain diagnosis should not be coded in any case.

The corpus contains very short documents. For instance, the document

HISTORY: Left lower chest pain. Rule-out

pneumonia. IMPRESSION: Normal chest.

has one label *786.50 (cough)* as *486 (pneumonia)* is ruled out.

The main conclusion of the shared task in 2007 was that simple rule-based systems generally outperform bag-of-words-based machine learning models. The rules were extracted from ICD guidelines and/or from the training corpus using simple statistical measures, then they were checked or extended manually. Several systems of the challenge employed a negation and speculation detection submodule. The (manually highly fine-tuned) top systems of the CMC shared task achieved an F-measure of 88-89 (Pestian et al., 2007; Farkas and Szarvas, 2008).

The I2B2 Obesity Dataset was also the subject of a clinical natural language processing shared task. The challenge in 2008 focused on analyzing clinical discharge summary texts and addressed the following question: "Who is obese and what co-morbidities do they have?" (Uzuner, 2009). Target diseases (document labels) included obesity and its 15 most frequent co-morbidities exhibited by patients. In our experiments, we used the same train/evaluation split as that of the shared task. Here a special aspect of the corpus is that the documents are semi-structured, i.e. they contain headings like *discharge medications* and *admit diagnosis*. By pasting the given heading to the beginning of each sentence, we incorporated it into the local context. The top performing systems of the shared task employed mostly hand-crafted rules for indicator selection and for negation and uncertainty detection as well. They achieved an F-measure[1] of 96-97 (Uzuner, 2009; Solt et al., 2009).

Wikipedia Soccer Dataset. We constructed a corpus based on Wikipedia articles and categories[2]. The categories assigned to Wikipedia articles can be regarded as labels (for example, the labels of *David Beckham* in the Wikipedia are *English people*, *expatriate soccer player*, *male model* and *A.C. Milan player*, *Manchester United player*). Based on the categories of Wikipedia, classifiers can be trained to tag unlabeled texts or even add missing category assignments to Wikipedia (Schönhofen, 2006).

For a case study we focused on learning English soccer clubs that a given sportsman played for. Note that this task is an information-oriented document labeling task as the clubs for which a sportsman played are usually just mentioned (especially for smaller clubs) in the article of a player. The Wikipedia category *Footballers in England by club* contains 408 subcategories (for the present and past). We selected the best known clubs (where the category label for the club is assigned to more than 500 player pages). Each article referring to a player having a category assignment to these clubs was downloaded and the textual parts were extracted. Then a random 3:1 train:evaluation split of the document set was used.

4 Document-labeling with CSD

We introduce here an iterative solution which selects indicator phrases and trains a content shift detector at the same time. Our focus will be on *multi-label document classification* tasks where multiple class labels can be assigned to a single document. In this study we will not deal with the modeling of inter-label dependencies, so binary (positive versus negative) and multi-class document classifications (where exactly one label has to be assigned to a single document) can be regarded as special cases of this multi-label classification problem. Our resulting multi-label model is then a set of binary classifiers – "assign a label" classifiers for each class label – and the final prediction on a document is simply the union of the labels forecasted by the individual classifiers.

Our key assumption in the multi-label environment is that while indicator phrases have to be selected on a per class basis, the content shifters can be learnt in a class-independent (aggregated) way i.e. we can assume that within one task, each class label belongs to a given semantic domain (determined by the task), thus the content shifters for their indicator phrases are the same. This approach provides an adequate amount of training samples for content shift detector learning.

[1] Using the definitions of the challenge, the evaluation metric applied here is the micro F-measure of the textual task on the YES versus every other class.

[2] The dataset is available as the supplementary material.

Table 2: Example feature representation of local contexts of *Arsenal*. The prefix NP stands for the lemma features from the deepest noun phrase; D,DR and DEP marks the lemmas, roles and their combination in the dependency path, respectively; SUBJ and SUBJD denote the lemmas and dependency roles on the "subject path", respectively.

His brother, Paul had a long career at Newcastle. (sentenceId=1, indicator=Newcastle)	
bag-of-word features	syntax-based features
he, brother, Paul, have, a, long, career, at	NP#a, NP#long, NP#career, NP#at D#career, D#have, DR#prepat, DR#dobj, DEP#career#prepat, DEP#have#dobj SUBJ#brother, SUBJ#Paul, SUBJ#he, SUBJD#he#poss
He was born in Gosforth, Newcastle and played for Arsenal. (sentenceId=2, indicator=Arsenal)	
bag-of-word features	syntax-based features
he, be, bear, in, Gosforth, Newcastle, and, play, for	D#play, DR#prepfor, DEP#play#prepfor SUBJ#he

4.1 Learning Content Shift Detectors

The key idea behind our approach is that a training corpus for task-specific content shifter learning can be automatically generated by exploiting the occurrences of indicators in various contexts. The local context of an indicator is assumed to have altered if it yields a false positive document-level prediction. More precisely, a training dataset can be constructed for learning a content shift detector in a way that the instances are the local contexts of each occurrence of indicator phrases in the training document set. The instances of this content shifter training dataset are then labeled as `non-altered` when the indicated label is among the gold-standard labels of the document in question or is labeled as `altered` otherwise. On this dataset, arbitrary binary classification models (S) can be trained.

As a feature representation of a local context of an indicator phrase, the bag-of-words of the sentence instance (excluding the indicator phrase itself) was used at the beginning. Our preliminary experiments showed that the tokens of the sentence after the indicator played a negligible role, hence we represented contexts just by tokens before the indicator.

Features concerning the syntactic context of the given indicator were also investigated. For this, we extended the feature set with features derived from the constituent and dependency parses of the sentence[3]. First, the deepest noun phrase which includes the indicator phrase was identified, then all lemmas from its subtree were gathered. From the dependency parse, the lemmas and dependency labels on the directed path from the indicator to the root node (main path) were extracted. The directed paths branching from this main path starting with `subject` dependency were also used for feature extraction (note that these walk in opposite direction to that of the main path). The intuition of the latter was that the subject of the given information – as it can differ from the target entity of extraction – is of great importance. We note that we recognize the in-sentence subject and employing a co-reference module would probably increase the value of these features.

Table 2 exemplifies the feature representation of local contexts of the *Newcastle* and *Arsenal* indicators for the Wikipedia soccer task. In both sentences, a naive system would extract *Newcastle* as false positives. We want to learn content shifters from them along with the true positive match of *Arsenal* in sentence 2. From the first example the CSD could learn even that the bag-of-word contains *brother* or the `SUBJ=`*brother*. However, in the second example, the bag-of-word representation is not sufficient to learn that the local context of *Newcastle* is `altered` because it is the subset of the bag-of-word representation of *Arsenal*'s `non-altered` local context. In this case the syntactic context representation can help and in our CSD `DEP=`*play#prepfor* gets high weight for the `non-altered` class.

[3]We parse only the sentences which contain indicator phrase which makes these features computable in reasonable time even on bigger document sets.

4.2 Co-learning of Indicator Selection and CSD

If document labels are available at training time, an iterative approach can be used to learn the local content shift detector and the indicator phrases as well. The training phase of this procedure (see Algorithm 1) has two outputs, namely the set of indicator phrases for each label I and the content shift detector S which is a binary function for determining whether the sense of an indicator in a particular local context is being altered. Good indicator phrases are those that identify the class label in question when they are present. In each step of the iteration we select indicator phrases $I[l]$ for each label l based on the actual state of the document set D'. Using these $I[l]$s we train a CSD S. Then we apply it to the original dataset D and we delete each local context from the documents which was predicted to be altered by S.

Algorithm 1 Co-learning of labels and CSD

Input: L class labels, D labeled training documents
  $D' \leftarrow D$
  repeat
    for all $l \in L$ **do**
      $I[l] \leftarrow$ indicatorSelection(D', l)
    end for
    $S \leftarrow$ learnCSD(D', I)
    $D' \leftarrow$ removeAlteredParts(D, S)
  until convergence
  return I, S

The indicator selection and content shifter learning phases can form an iterative process. The better the selected indicators are, the better the content shift detectors can be learnt. By applying the content shift detector to each token of the documents, each part of the texts lying within the scope of a content shifter can be removed[4]. By using such a cleaned training document set (D'), better indicators can be selected. These steps can be repeated until some convergence criterion is reached. In our experiments we simply used a fixed iteration number to gain an insight into the behavior of the approach.

[4]In our first experiments introduced here, we removed the parts of the documents classified as altered. Instead of removing these parts they may be marked and then different features may be extracted from them.

Algorithm 2 Document labeling with CSD

Input: d document, I indicator sets, S CSD
  pred $\leftarrow \emptyset$
  for all $l \in L$ **do**
    for all $o \in$ occurrences($d, I[l]$) **do**
      if not altered(o, S) **then**
        pred $\leftarrow$ pred $\cup\, l$
      end if
    end for
  end for
  return pred

The prediction procedure of the approach (see Algorithm 2) then looks for occurrences of the indicator phrases in the text and checks whether they are altered in a certain local context. A non-altered indicator directly assigns a class label without any global consistency check on assigned labels.

We note here, that the local relationship among tokens (i.e. the local context) may be taken into account by incorporating this information directly into the feature space of a document classifier (as an alternative of our co-learning procedure), but the number of features would exponentially increase and submodels for each indicator phrases should be learnt which would made such a classification task intractable.

4.3 Indicator Phrase Selection

Indicator phrases are sequences of tokens whose presence implies the positive class. We aimed to extract phrases with the length of 1,2 or 3 (and we used exact matching after lemmatisation). There are several possible ways of developing indicator selection algorithms. One way is to treat it as a special feature selection procedure where the goal is to select a set of features (uni-, bi-, trigrams of a bag-of-word model) which achieves high recall along with moderate precision as false positives are expected to be eliminated by the local CSD in our two-step approach. Indicator selectors can be even derived from most classifiers which are based on feature weighting (like MaxEnt and AvgPerceptron) or feature ranking (like rule-based classifiers)[5] as well. However indicator selection is not the focus of this

[5]A derivation is more complicated or unfeasible for example-based classifiers like SVMs.

Table 3: Results obtained for local content shift detection in a precision/recall/F-measure format.

		CMC	Obesity	Soccer
Trained	BoW	90.7 / 60.7 / 72.7	82.1 / 35.4 / 49.4	75.0 / 70.6 / 72.7
	BoW+syntactic	88.3 / 60.2 / 71.6	84.4 / 33.3 / 47.8	81.0 / 78.9 / 79.9
Hand-crafted	CSSDB	94.7 / 53.3 / 68.2	42.0 / 57.9 / 48.7	36.8 / 9.8 / 15.5
	in-sentence	80.7 / 65.2 / 72.2	70.5 / 40.5 / 51.5	N/A

work.

For our experiments, a feature evaluation-based greedy algorithm was employed to select the set of indicators from the pool of token uni- and bigrams. The aim of the the indicator selection here is to cover each positive documents while introducing a relatively small amount of false positives. The greedy algorithm iteratively selects the 1-best phrase according to a feature evaluation metric based on the actual state of covered documents and adds it to the indicator phrase set. The process is iterated while the score – in terms of the applied feature evaluation metric – of the 1-best phrase is above a threshold t. The quality of the selected indicator set is highly dependent on the stopping threshold t, but as individual feature evaluation functions are very fast and the number of good indicators is usually low (4-5), the whole greedy indicator selection is fast, hence t can be fine-tuned without overfitting on the training sets employing a cross-validation procedure. As a feature evaluation metric we employed $p(+|f)$ the probability of the positive class "+" conditioned on the presence of a feature f because preliminary experiments did not show any significant advances for more complex metrics.

5 Experiments

Experiments were carried out on the three datasets introduced in Section 3 with local content shift detection as an individual task and also to investigate its added value to information-oriented document labeling.

In our experiments, we applied the sentence splitter and lemmatizer implementation of the MorphAdorner package[6] and the Stanford tokenizer and lexicalized PCFG parser (Klein and Manning, 2003)[7].

5.1 Content Shifter Learning Results

In order to evaluate content shift detection as an individual task, a set of indicator phrases have to be fixed as an input to the CSD. We used manually collected indicator phrases for each label for each dataset. We utilized the terms of Farkas and Szarvas (2008) and Farkas et al. (2009) collected for the CMC and Obesity datasets, respectively and club names for the Soccer dataset in our first branch of experiments. Note that the clinical term sets here have been manually fine-tuned as they were developed for participating systems of the shared tasks of the corpora.

Based on the occurrences of these fixed indicator phrases, CSD **train**ing datasets were built from the local contexts of the three datasets and binary classification was carried out by using MaxEnt. Table 3 shows the results achieved by the learnt CSDs using the bag-of-word feature representation (row 1) along with the ones obtained by the feature set that was extended with syntactic patterns (raw 2). Here, the precision/recall/F-measure values measure how many false positive matches of the indicator phrases can be recognized (the F-measure of the `altered` class), i.e. here, the true positives are local contexts of an indicator phrase which do not indicate a document label in the evaluation set and the local content shift detector predicted it to be `altered`.

For comparison purposes, we employed **manually developed CSDs** which were fine-tuned for the medical shared task datasets. Row 3 of Table 3 (we refer to it as *content shifted sentence detection baseline (CSSDB)* later on) shows the results archived by the method which predicts every sentence to be `altered` which contain any *cue phrases* for negation, modality and different experiencer. Note that off-the-shelf tools are available just for these types of content shifters. We collected cue phrases for such a content shifted sentence detection from the

[6] `morphadorner.northwestern.edu/`

[7] The JAVA implementation of the entire framework and dataset adapters can be found as the supplementary material

works of Chapman et al. (2007), Light et al. (2004) and Vincze et al. (2008) and from the experiments of Farkas and Szarvas (2008) and Farkas et al. (2009).

For the CMC and Obesity tasks, hand-crafted in-sentence CSDs were also available (Farkas and Szarvas, 2008; Farkas et al., 2009), i.e. they apply heuristics – which usually tries to recognise clause boundaries – for determining the scope of a negaton/modality cue. This CSD is more fine-grained than the sentence-level one as here a part of a sentence can be detected as `altered` while other parts as `non-altered`. The results of these detectors – two different CSDs, both highly fine-tuned for the corresponding shared task – are listed in the last row of Table 3.

On the CMC dataset, our machine learning approach identified mostly negation and speculation expressions as **content shifters**; the top weighted features for the positive class of the MaxEnt model were *no, without, may* and *vs*. They can filter out false positive matches like

Hyperinflated without focal pneumonia..

On the Obesity dataset, similar content shifters were learnt along with references to family members (like the terms *mother* and *uncle*, and the *family history* header). The significance of these types of content shifters may be illustrated by the following sentence:

History of hypertension in mother and sister.

The soccer task highlighted totally different content shifters which is also the reason for the poor performance of CSSDB. The mention of a club name which the person in question did not play for (false positives) is usually a rival club, club of an unsuccessful negotiation or club which was managed by the footballer after his retirement. For example:

His last game was against Chelsea at Stamford Bridge.

He was a coach at United during his son's playing career.

Summing up, the machine learnt CSDs proved to be competitive with the manually fine-tuned CSD on the three datasets. Table 3 shows that learnt CSDs were able to eliminate a significant amount of false positive indicator phrase matches on each of the three datasets. The hand-crafted CSDs developed for the medical texts certainly work poorly (an F-score of 15.5) on the Soccer dataset as content shifters different from negation, hedge and experiencer are useful there. On the other hand, the content shifters could be learnt on this dataset by our CSD approach (achieving F-score of 79.9). In the clinical corpora, the features from the syntactic parses just confused the system, but they proved to be useful on the Soccer corpus. Here, the dependency parse achieved improvements in terms of both precision and recall (the number of true positives increased by 137) which can be mainly attributed to the prepositions *against* and *over*. The reason why it did not advance on the clinical corpora is probably the domain difference between the training corpus of the parsers and the target texts, i.e. the parsers trained on the Wall Street Journal could not build adequate dependency parses on clinical notes.

As a final comparison we investigated the manually annotated BioScope corpus (Vincze et al., 2008) as a CSD. The CMC corpus is included in the BioScope corpus where text spans in the in-sentence scope of speculation and negation were annotated. We used this manual annotation as an oracle CSD and got an F-measure of 75.2 (which is significantly higher than the scores 72.2 and 72.7 archived by the hand-crafted and trained CSD respectively). This score can be regarded as an upper bound for the amount of false positive indicator matches that can be fixed by local speculation and negation detectors. The remaining false positives are not covered by the linguistically motivated annotations of BioScope, i.e. false positives recognizable by domain knowledge (e.g. coding symptoms should be omitted when a certain diagnosis that is connected with the symptom in question is present in the document) are not marked.

Our **error analysis** revealed that most of the errors of the learnt CSDs is due to the lack of semantic link between lexical units. For instance, on the Soccer dataset it could learn that the token *coach* occuring in the sentence in question indicates an `altered` content, but it was not able to recognise this for *trainer*. The reason for that is simple, the ratio of occurrences of *trainer*:*coach* is 5:95 in the

Table 4: Results obtained by document multi-labeling algorithms in a precision/recall/F-measure format.

rowID				CMC	Obesity	Soccer
1	Baseline	SVM	with CSSDB	87.7 / 76.7 / 81.8	90.0 / 81.3 / 85.4	92.2 / 75.1 / 82.8
2		MaxEnt		92.2 / 72.2 / 81.0	91.4 / 87.6 / 89.4	92.2 / 77.4 / 84.2
3		PART		83.9 / 80.6 / 82.2	87.3 / 86.4 / 86.8	81.2 / 77.0 / 79.0
4	Indicator Selection	without CSD		78.0 / 85.1 / 81.4	89.2 / 93.6 / 91.3	84.4 / 83.7 / 84.1
5		with CSSDB		79.0 / 84.1 / 81.4	94.8 / 86.6 / 91.1	85.2 / 85.5 / 85.3
6		with learnt CSD		83.1 / 83.2 / 83.2	91.7 / 92.9 / 92.3	91.7 / 85.2 / **88.3**
7		after 3 iterations		82.4 / 86.8 / **84.6**	92.6 / 95.4 / **94.0**	92.5 / 84.0 / 88.0
8		after 10 iterations		82.4 / 86.8 / 84.6	92.7 / 95.4 / 94.0	92.5 / 84.0 / 88.0
9	Baseline	MaxEnt with learnt CSD		89.9 / 77.0 / 83.0	91.9 / 90.4 / 91.1	95.0 / 78.7 / 86.1

training corpus. Increasing the training size may be a simple way to overcome this shortcoming. Note that increasing the number of labels (e.g. introducing more soccer clubs in the Soccer task) would also directly increase the size of training dataset as we use the occurrences of the indicator phrases belonging to each of the labels for training a CSD. The solution for the rare cases would require the explicit handling of semantic relatedness (by utilising existing semantic resources or trying to automatically identify task-specific relations).

5.2 Document Labeling Results

The second branch of experiments investigated the added value of CSDs in information-oriented document labeling tasks. Table 4 summarizes the results we got on the three datasets using the micro-averaged $F_{\beta=1}$ of assigned labels (positive class).

As **baseline** systems we trained binary SVMs with a linear kernel, MaxEnts and PARTs – a rule-learner classification algorithm (Frank and Witten, 1998) – for each label using the bag-of-word representation of the documents (implementations of SVMligth (Joachims, 1999), MALLET (McCallum, 2002) and WEKA (Witten and Frank, 1999) were used). The first two learners are popular choices for document classification, while the third is similar to our simple indicator selection procedure. We did not tuned the parameters of the classifiers, we used the default ones everywhere.

To have a fair comparison, we applied to preprocessing steps on dataset of these document classifiers. First, we removed from the training and evaluation raw documents which were predicted to be `altered` by CSSDB. Second, as our indicator selection phrase can be regarded as a special feature selection method, we carried out an Information Gain-based feature selection (keeping the 500 best-rated features proved to be the best solution) on the bag-of-word representation of the documents. The effect of these two preprocessing steps varied among datasets. It improved the F-score of the MaxEnt baseline document classifier by 20%, 2% and 3% on the Obesity, CMC and Soccer datasets, respectively (the F-measures of Table 4 are the values we got by employing pre-processing).

The **indicator selection** results presented in the rows 4-8 of Table 4 made use of the $p(+|f)$-based indicator selector with a five-fold-cross-validated stopping threshold t (introduced in Section 4.3). Row 4 contains the results of using the selected indicators without any CSD. Indicator selection with the CSSDB was applied for the 5th row. Rows 6-8 of Table 4 show the results obtained after one, three and ten iterations of the full learning algorithm (see Algorithm 1). For training the CSD, we employed MaxEnt as a binary classifier for detecting `altered` local contexts and we used the basic BoW feature representation for the clinical tasks while the extended (BoW+syntactic) one for the Soccer dataset.

In the final experiment (the last row of Table 4)) we investigated whether the learnt content shift detector can be applied as a general "**document cleaner**" tool. For this, we trained the baseline MaxEnt document classifier with feature selection on documents from which the text spans predicted to be `altered` by the learnt CSD in the tenth iteration were removed. This means that the systems

used in row 2 and row 9 differ only in the applied document cleaner pre-processing steps (the first one applied the CSSDB while the latter one employed the the learnt CSD).

The difference between the best baseline and the indicator selector with learnt CSD and between the best baseline and the document classifier with learnt CSD were statistically significant[8] on each dataset. The difference between the predictions after the 1st and 3rd iterations were statistically significant on the CMC and the Obesity corpora but not significant on the Soccer dataset. The difference between the 3th and 10th iterations were not significant in either case. Our co-learning method which integrated the document-labeling and CSD tasks significantly outperformed the baseline approaches – which use separate document cleaning and document labeling steps – on the three datasets.

On the clinical domains the automatically selected indicators were disease names, symptom names (e.g. *high blood pressure*), their spelling variants, synonyms (like *hypertension*) and their abbreviations (e.g *htn*). On the soccer domain club names, synonyms (like *The Saints*) and stadium names (e.g. *Old Trafford*) were selected. A label was indicated by 3-4 indicator phrases.

Note that in these information-oriented document multi-labeling tasks simple indicator selection-based document labelers alone achieved results comparable to the bag-of-words-based classifiers. The learnt content shift detectors led to an average improvement of 3.6% in the F-measure (i.e. a 24% error reduction). The effect of further iterations is various. As Table 4 shows, three iterations brought an increase on the CMC and Obesity datasets but not on the Soccer corpus. After a few iterations the set of indicator phrases and the content shift detector did not change substantially. The results achieved by the MaxEnt document classifier employing the ”cleaned” training documents (last row of Table 4) are significantly better (an average improvement of 1.9% in the F-measure and 12% error reduction) than those by the CSSDB (row 2) but the indicator selector approach performed even better.

[8] According to McNemar's test with P-value of 0.001

6 Conclusions

In this paper, we dealt with information-oriented document labeling tasks and investigated machine learning approaches for local content shift detectors from document-level labels. We demonstrated experimentally that a significant amount of false positive matches of indicator phrases can be recognized by trained content shift detectors. Our trained CSD does not use any task or domain specific knowledge and exploits the false and true positive matches of indicator phrases, i.e. it uses only document-level annotation. This task-independent approach achieved competitive results with CSDs which were manually fine-tuned for particular datasets. The empirical results also support the idea of generalized local CSD (false positive removal) opposite to developing independent CSD for particular language phenomena (like negation and speculation).

A co-learning framework for training local content shift detectors and indicator selection was introduced as well. Our method integrates document classification and CSD learning, which are traditionally used as independent submodules of applications. Experiments on three information-oriented document-labeling datasets – from two application areas – with simple indicator selection and syntactic parse-based content shifter learning were performed and the results show a clear improvement over the bag-of-word-based document classification baseline approaches.

However, the proposed content shift detector learning approach is tailored for information-oriented document labeling tasks, i.e. it performs well when not too many and reliable indicator phrases are present. In the future, we plan to investigate and extend the framework for the general document classification task where many indicators with complex relationships among them determine the labels of a document but local content shifters can play an important role.

Acknowledgements

This work was partially founded by the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences. Richárd Farkas was also funded by Deutsche Forschungsgemeinschaft grant SFB 732.

References

Eiji Aramaki, Yasuhide Miura, Masatsugu Tonoike, Tomoko Ohkuma, Hiroshi Mashuichi, and Kazuhiko Ohe. 2009. TEXT2TABLE: Medical Text Summarization System Based on Named Entity Recognition and Modality Identification. In *Proceedings of the BioNLP 2009 Workshop*, pages 185–192.

Wendy W. Chapman, David Chu, and John N. Dowling. 2007. Context: an algorithm for identifying contextual features from clinical text. In *Proceedings of the ACL Workshop on BioNLP 2007*, pages 81–88.

Richárd Farkas and György Szarvas. 2008. Automatic construction of rule-based icd-9-cm coding systems. *BMC Bioinformatics*, 9(Suppl 3):S10.

Richárd Farkas, György Szarvas, István Hegedüs, Attila Almási, Veronika Vincze, Róbert Ormándi, and Róbert Busa-Fekete. 2009. Semi-automated construction of decision rules to predict morbidities from clinical texts. *Journal of the American Medical Informatics Association*, 16:601–605.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The CoNLL-2010 Shared Task: Learning to Detect Hedges and their Scope in Natural Language Text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010): Shared Task*, pages 1–12.

Eibe Frank and Ian H. Witten. 1998. Generating accurate rule sets without global optimization. In *Proc. of Fifteenth International Conference on Machine Learning*, pages 144–151.

Carol Friedman, Philip O. Alderson, John H. M. Austin, James J. Cimino, and Stephen B. Johnson. 1994. A General Natural-language Text Processor for Clinical Radiology. *Journal of the American Medical Informatics Association*, 1(2):161–174.

Viola Ganter and Michael Strube. 2009. Finding Hedges by Chasing Weasels: Hedge Detection Using Wikipedia Tags and Shallow Linguistic Features. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 173–176.

Thorsten Joachims, 1999. *Making large-scale support vector machine learning practical*, pages 169–184. MIT Press, Cambridge, MA, USA.

Halil Kilicoglu and Sabine Bergler. 2009. Syntactic Dependency Based Heuristics for Biological Event Extraction. In *Proceedings of the BioNLP Workshop Companion Volume for Shared Task*, pages 119–127.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 Shared Task on Event Extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st ACL*, pages 423–430.

Leah S. Larkey and W. Bruce Croft. 1995. Automatic assignment of icd9 codes to discharge summaries. Technical report.

Marc Light, Xin Ying Qiu, and Padmini Srinivasan. 2004. The language of bioscience: Facts, speculations, and statements in between. In *Proc. of Biolink 2004, Linking Biological Literature, Ontologies and Databases (HLT-NAACL Workshop:)*, pages 17–24.

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

Ben Medlock and Ted Briscoe. 2007. Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the ACL*, pages 992–999, June.

Roser Morante, V. Van Asch, and A. van den Bosch. 2009. Joint memory-based learning of syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*, pages 25–30.

Arzucan Özgür and Dragomir R. Radev. 2009. Detecting Speculations and their Scopes in Scientific Text. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1398–1407.

John P. Pestian, Chris Brew, Pawel Matykiewicz, DJ Hovermale, Neil Johnson, K. Bretonnel Cohen, and Wlodzislaw Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Proceedings of the ACL Workshop on BioNLP 2007*, pages 97–104.

Roser Sauri and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227–268.

Peter Schönhofen. 2006. Identifying document topics using the wikipedia category network. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 456–462.

Illés Solt, Domonkos Tikk, Viktor Gál, and Zsolt Tivadar Kardkovács. 2009. Semantic classification of diseases in discharge summaries using a context-aware rule-based classifier. *J. Am. Med. Inform. Assoc.*, 16:580–584, jul.

Stephanie Strassel, Mark A. Przybocki, Kay Peterson, Zhiyi Song, and Kazuaki Maeda. 2008. Linguistic resources and evaluation techniques for evaluation of cross-document automatic content extraction. In *LREC*.

György Szarvas. 2008. Hedge Classification in Biomedical Texts with a Weakly Supervised Selection of Keywords. In *Proceedings of ACL-08*, pages 281–289.

O. Uzuner, Ira Goldstein, Yuan Luo, and Isaac Kohane. 2008. Identifying Patient Smoking Status from Medical Discharge Records. *Journal of American Medical Informatics Association*, 15(1):14–24.

Ozlem Uzuner. 2009. Recognizing obesity and comorbidities in sparse data. *Journal of American Medical Informatics Association*, 16(4):561–70.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope Corpus: Biomedical Texts Annotated for Uncertainty, Negation and their Scopes. *BMC Bioinformatics*, 9(Suppl 11):S9.

Ian H. Witten and Eibe Frank. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Collaborative Ranking: A Case Study on Entity Linking

Zheng Chen
Computer Science Department
Graduate Center
City University of New York
zchen1@gc.cuny.edu

Heng Ji
Computer Science Department
Queens College and Graduate Center
City University of New York
hengji@cs.qc.cuny.edu

Abstract

In this paper, we present a new ranking scheme, collaborative ranking (*CR*). In contrast to traditional non-collaborative ranking scheme which solely relies on the strengths of isolated queries and one stand-alone ranking algorithm, the new scheme integrates the strengths from multiple collaborators of a query and the strengths from multiple ranking algorithms. We elaborate three specific forms of collaborative ranking, namely, micro collaborative ranking (MiCR), macro collaborative ranking (MaCR) and micro-macro collaborative ranking (MiMaCR). Experiments on entity linking task show that our proposed scheme is indeed effective and promising.

1 Introduction

Many natural language processing tasks can be formalized as a ranking problem, namely to rank a collection of candidate "objects" with respect to a "query". For example, intensive studies were devoted to parsing in which multiple possible parsing trees or forests are ranked with respect to a sentence (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008), machine translation in which multiple translation hypotheses are ranked with respect to a source sentence (Och, 2002; Shen et al., 2005), anaphora resolution in which multiple antecedents are ranked with respect to an anaphora (Yang et al., 2008), and question answering in which multiple possible answers are ranked with respect to a question (Ravichandran et al., 2003). Previous studies mainly focused on improving the ranking performance using one stand-alone learning algorithm on isolated queries.

Although a wide range of learning algorithms (unsupervised, supervised or semi-supervised) is available, each with its strengths and weaknesses, there is not a learning algorithm that can work best on all types of data. In such a situation, it would be desirable to build a "collaborative" model by integrating multiple models. Such an idea forms the basis of ensemble methodology and it is well-known that ensemble methods (e.g., bagging, boosting) can improve the performance of many problems, in which classification is the most intensively studied (Rokach, 2009). The other situation is related with isolated queries handled by learning algorithms. The single query may not be formulated with the best terms or the query itself may not contain comprehensive information required for a high-performance ranking algorithm. Therefore, techniques of query expansion or query reformulation can be introduced and previous research has shown the effectiveness of those techniques in such applications as information retrieval and question answering (Manning et al., 2008; Riezler et al., 2007). Nevertheless, previous research normally considers query reformulation as a new query for the ranking system, it would be more desirable to form a larger-scale "collaborative" group for the query and make a unified decision based on the group.

Inspired from human collaborative learning in which two or more people form a group and accomplish work together, we propose a new ranking scheme, *collaborative ranking*, which aims to imitate human collaborative learning and enhance system ranking performance. The main idea is to seek collaborations for each query from two levels:

(1) query-level: search a group of query collaborators, and make the joint decision from the group together with the query using a stand-alone ranking algorithm.

(2) ranker-level: design a group of multiple rankers, and make the joint decision from the entire group on a single query.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 771–781,
Edinburgh, Scotland, UK, July 27–31, 2011.

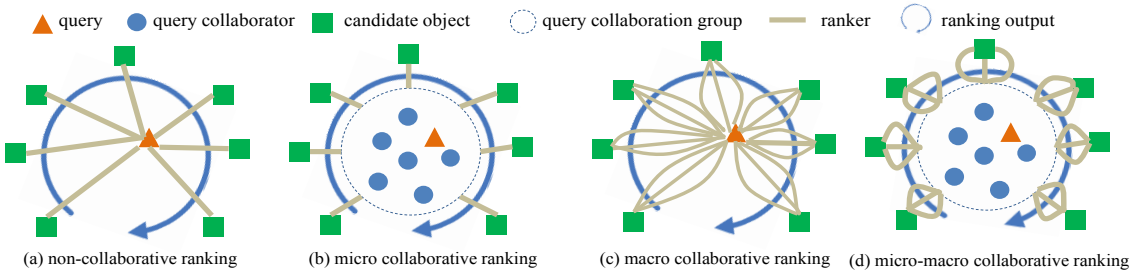


Figure 1: Non-collaborative ranking and three collaborative ranking approaches.

Figure 1 presents an intuitive illustration of four ranking approaches, including the traditional non-collaborative ranking and three collaborative ranking forms: micro collaborative ranking (MiCR), macro collaborative ranking (MaCR), and micro-macro collaborative ranking (MiMaCR).

Compared with the traditional non-collaborative ranking that only leverages the information contained in a single query and only applies one ranking function (Figure 1 (a)), the three collaborative ranking approaches have the following advantages:

(1)MiCR (corresponding to query-level collaboration[1]) leverages the information contained in the collaborators of a query. Figure 1 (b) demonstrates that 6 query collaborators together with the query form a query collaboration group.

(2)MaCR (corresponding to ranker-level collaboration[2]) integrates the strengths from two or more rankers. Figure 1 (c) demonstrates an example of 3 rankers.

(3)MiMaCR combines the advantages from MiCR and MaCR as shown in Figure 1 (d).

In this paper, we will show the efficacy of collaborative ranking on the entity linking task defined in the Knowledge Base Population (KBP) track (Ji et al., 2010) at Text Analysis Conference (TAC). Each query in the task is associated with a name string and its context document. Traditional approaches for entity linking only made use of the lexical or document level information contained in the query, however, it may not be sufficient for the task. The intuition why query-level collaboration may work is that it leverages more comprehensive information about the entity mention from multiple "collaborators" (related documents containing the name string). Furthermore, previous work on this task mainly focused on comparing one ranking algorithm with the others, however, each ranking algorithm has its own strengths, and therefore, ranker-level collaboration can potentially improve the performance. Last, the combination of query-level and ranker-level collaboration can lead to further performance gains.

[1]Query is normally expressed by small-scale data structure, so called micro.

[2]Ranker is normally implemented by large-scale algorithm, so called macro.

2 Non-collaborative Ranking

Let q denote a query. Let $o^{(q)} = \left\{o_1^{(q)}, \ldots, o_{n^{(q)}}^{(q)}\right\}$ denote the object set associated with q, where $n^{(q)}$ denotes the size of the $o^{(q)}$. The goal of non-collaborative ranking is to seek a ranking function f such that it computes ranking scores for the candidates in the object set, i.e., $y^{(q)} = f(o^{(q)}) = \left\{y_1^{(q)}, \ldots, y_{n^{(q)}}^{(q)}\right\}$.

Earlier studies on non-collaborative ranking mainly explored unsupervised approaches, e.g., vector space model, link based algorithm such as PageRank (Page et al., 1998). Unsupervised approaches are based on well-established statistical and probability theory, nevertheless, they suffer from some drawbacks, for example, it is hard to tune parameters. Recently, supervised approaches (named "learning to rank") that automatically learn ranking functions from training data become the focus of ranking research. In the literature, supervised approaches are categorized into three classes, namely, pointwise, pairwise, and listwise. We summarize a comparison of the three approaches in Table 1. We use the following notations in the table.

Let $\mathcal{Q} = \{q_1, \ldots, q_N\}$ denote the set of N queries in the training data, each query q_i is associated with a set of objects $o^{(q_i)} = \left\{o_1^{(q_i)}, \ldots, o_{n^{(q_i)}}^{(q_i)}\right\}$ and a set of ground-truth ranking scores $y^{(q_i)} =$

	pointwise	pairwise	listwise
approach overview	common: 1) use training samples; 2) learn the best ranking function by minimizing a given loss function; 3) apply the ranking function at ranking step		
	transform ranking to regression or classification on single objects	transform ranking to classification on object pairs	ranking by learning from lists of objects
training set	$\{\left(x_j^{(q_i)}, y_j^{(q_i)}\right)\}_{j=1,\ldots,n^{(q_i)};\ i=1,\ldots,N}$	$\{\left(x_j^{(q_i)}, x_k^{(q_i)}, y\right)\}_{j=1,\ldots,n^{(q_i)};k=1,\ldots,n^{(q_i)};\ k\neq j;\ i=1,\ldots,N}$ $y=\begin{cases}+1\ if\ x_j^{(q_i)} > x_k^{(q_i)}\\ -1\ if\ x_j^{(q_i)} \leq x_k^{(q_i)}\end{cases}$	$\{(x^{(q_i)}, y^{(q_i)})\}_{i=1,\ldots,N}$
loss function	pointwise loss, e.g., square loss(Chen et al., 2009)	pairwise loss, e.g., hinge loss(Zhang, 2004), exponential loss(Bartlett et al., 2003), logistic loss(Lin, 2002)	listwise loss, e.g., cross entropy loss(Cao et al., 2007),cosine loss(Qin et al., 2007)
pros and cons	pros: classification is well studied cons:1) only consider one object at a time ignoring relationship among objects	pros: classification is well studied cons:1) only consider pairwise orders; 2) biased towards lists with more objects	pros: fully consider relationship among objects cons: 1) less well studied in theory
selected algorithms	Discriminative model for IR (Nallapati, 2004); McRank (Li et al., 2007)	SVM Ranking(Joachims, 2002); RankBoost(Freund et al., 2003); RankNet(Burges et al., 2005)	ListNet (Cao et al., 2007); RankCosine(Qin et al., 2007); ListMLE (Xia et al., 2008)

Table 1: Comparison of pointwise, pairwise and listwise ranking approaches.

$\left\{y_1^{(q_i)}, \ldots, y_{n^{(q_i)}}^{(q_i)}\right\}$. Let $x_j^{(q_i)} = \varphi(q_i, o_j^{(q_i)})$ denote a feature vector associated with each query-object pair $(q_i, o_j^{(q_i)})$.

3 Collaborative Ranking

3.1 Micro Collaborative Ranking(MiCR)

Micro collaborative ranking is characterized by integrating joint strengths from multiple query collaborators and the query itself. It is based on the following assumptions:

• Expandability: Query is expandable, that is, it is able to find potential collaborators.

• Redundancy: Collaborators and query may share redundant information.

• Diversity: Collaborators exhibit multifaceted information that may complement the information contained in the query.

• Robustness: Noisy collaborators are allowable, and they could be put under control.

Let $cq^{(q)} = \{cq_1, \ldots, cq_k\}$ be the k collaborators of a query q. For each object $o_j^{(q)}$ associated with q, we form $k+1$ feature vectors $x_j^{(q)} = \varphi(q, o_j^{(q)}), x_j^{(cq_1)} = \varphi(cq_1, o_j^{(cq_1)}), \ldots, x_j^{(cq_k)} = \varphi(cq_k, o_j^{(cq_k)})$. Let f be a ranking function which is obtained by either an unsupervised or supervised approach. There are two important steps that distinguish MiCR from traditional non-collaborative ranking approaches:

• Step (1): searching the best k collaborators of q.

• Step (2): simulating the interaction of k collaborators at the ranking step.

Solutions for step (1) can vary from case to case. In our case study presented later, we transform the collaborator searching problem into a clustering problem. Collaborators of a query are then formed by members (excluding the query) in a cluster which contains the query and k is the size of the cluster minus one.

We transform the problem of step (2) into solving a function g_1 such that a ranking score $y_j^{(q)}$ can be computed for each object $o_j^{(q)}$. One approach to computing g_1 is to firstly compute the ranking scores of collaborators and query using the ranking function f and then combine those ranking scores in some way (Formula 1). The other approach is to learn a supervised ranking function f' which takes collaborators and query as input (Formula 2).

$$y_j^{(q)} = g_1(f\left(x_j^{(q)}\right), f\left(x_j^{(cq_1)}\right), \ldots, f\left(x_j^{(cq_k)}\right)) \quad (1)$$

$$y_j^{(q)} = g_1(\bullet) = f'\left(x_j^{(q)}, x_j^{(cq_1)}, \ldots, x_j^{(cq_k)}\right) \quad (2)$$

We present three specific forms of g_1 in Formula 1, namely, max, min, and weighted. We can also define a special case of weighted, called "average" in which $w_0 = w_1 \ldots = w_k = 1/(k+1)$.

- max: $y_j^{(q)} = \max(f\left(x_j^{(q)}\right), \ldots, f\left(x_j^{(cq_k)}\right))$
- min: $y_j^{(q)} = \min(f\left(x_j^{(q)}\right), \ldots, f\left(x_j^{(cq_k)}\right))$
- weighted: $y_j^{(q)} = w_0 f\left(x_j^{(q)}\right) + \sum_{i=1}^{k} w_i f\left(x_j^{(cq_i)}\right)$

We will discuss three supervised versions of g_1 (Formula 2) in section 4.4. A general algorithm for MiCR is presented in Algorithm 1.

Algorithm 1 MiCR Algorithm.

Input:
a query q; a set of objects $o^{(q)}$; a function g_1
Output:
a set of ranking scores $y^{(q)}$

```
Search k collaborators of q:
cq^(q) = {cq_1, ..., cq_k}.
for j = 1; j <= n^(q); j++ do
    Form k + 1 feature vectors: x_j^(q), x_j^(cq_1), ..., x_j^(cq_k).
    Compute function y_j^(q) = g_1(•).
end for
return y^(q)
```

3.2 Macro Collaborative Ranking(MaCR)

Macro collaborative ranking is characterized by integrating joint strengths from multiple rankers. It is based on the following assumptions:

- Independence: Each ranker can make its own ranking decisions.
- Diversity: Each ranker has its own strengths in making ranking decisions.
- Collaboration: Rankers in the group could collaborate to make a consensus decision under some mechanism.

Let $x_j^{(q)} = \varphi(q, o_j^{(q)})$ be the feature vector formed from the pair consisting of query q and an associated object $o_j^{(q)}$. Let $\mathcal{F}^* = \{f_1, \ldots, f_m\}$ be m existing ranking functions. We transform the computation of collaboration among rankers into solving the following composite function g_2:

$$y_j^{(q)} = g_2(f_1\left(x_j^{(q)}\right), \ldots, f_m\left(x_j^{(q)}\right)) \quad (3)$$

Similar with MiCR, g_2 can be expressed by max, min, weighted (average) respectively:

- max: $y_j^{(q)} = \max\{f_i\left(x_j^{(q)}\right)\}_{i=1}^{m}$
- min: $y_j^{(q)} = \min\{f_i\left(x_j^{(q)}\right)\}_{i=1}^{m}$
- weighted: $y_j^{(q)} = \sum_{i=1}^{m} w_i f_i\left(x_j^{(q)}\right)$

It is worth noting that max and min can be useful only if the ranking scores produced by various rankers can be compared to each other directly, however, in practice, this can hardly be true.

A special form of ranking problem is that only the best object is required as output. In this case, we have another version of g_2 which is called voting:

- voting: $y_j^{(q)} = \sum_{i=1}^{m} \text{sign}(f_i\left(x_j^{(q)}\right))$

in which $sign(\bullet)$ is an indicator function

$$sign(\bullet) = \begin{cases} 1 & \text{if } f_i \text{ outputs } o_j^{(q)} \text{ as the best object} \\ 0 & \text{otherwise} \end{cases}$$

A general algorithm for MaCR is presented in Algorithm 2.

Algorithm 2 MaCR Algorithm.

Input:
a query q; a set of objects $o^{(q)}$; a set of m ranking functions $\mathcal{F}^*$; a composite function g_2
Output:
a set of ranking scores $y^{(q)}$

```
for j = 1; j <= n^(q); j++ do
    Form a feature vector x_j^(q).
    Compute ranking scores: f_1(x_j^(q)), ..., f_m(x_j^(q)).
    Compute composite function: y_j^(q) = g_2(•).
end for
return y^(q)
```

3.3 Micro-Macro Collaborative Ranking (MiMaCR)

The above two ranking approaches can be further integrated into a joint model which is named Micro-Macro Collaborative Ranking (MiMaCR). In order to compute query-level and ranker-level collaboration jointly, we solve the following complex composite function g_3:

$$y_j^{(q)} = g_2(g_1(\bullet)) \quad (4)$$

in which, for each object $o_j^{(q)}$, firstly we compute m micro-ranking scores using m ranking functions on query-level collaborators:

$$m \begin{cases} g_1(f_1\left(x_j^{(q)}\right), f_1\left(x_j^{(cq_1)}\right), \ldots, f_1\left(x_j^{(cq_k)}\right)) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ g_1(f_m\left(x_j^{(q)}\right), f_m\left(x_j^{(cq_1)}\right), \ldots, f_m\left(x_j^{(cq_k)}\right)) \end{cases}$$

and secondly, we compute a macro-ranking score using g_2.

We can similarly define g_1 and g_2 as those in MiCR and MaCR. A general algorithm for MiMaCR is presented in Algorithm 3.

Algorithm 3 MiMaCR Algorithm.

Input:

a query q; a set of objects $o^{(q)}$; a set of ranking functions $\mathcal{F}^*$; functions g_1, g_2

Output:

a set of ranking scores $y^{(q)}$

1: Search k collaborators of q: $cq^{(q)} = \{cq_1, \ldots, cq_k\}$.
2: **for** $j = 1; j <= n^{(q)}; j++$ **do**
3: Form $k+1$ feature vectors: $x_j^{(q)}, x_j^{(cq_1)}, \ldots, x_j^{(cq_k)}$.
4: Compute m micro-ranking scores using $\mathcal{F}^*$ and g_1.
5: Compute the macro-ranking score using g_2.
6: **end for**
7: **return** $y^{(q)}$

4 A Case Study on Entity Linking

To demonstrate the efficacy of our collaborative ranking scheme, we apply it to the entity linking task defined in the TAC-KBP2010 program (Ji et al., 2010) because there is a large amount of training and evaluation data available and various non-collaborative ranking approaches have been proposed, as summarized in (McNamee and Dang, 2009; Ji et al., 2010).

4.1 Task Definition

The entity linking task aims to align a textual mention of a named entity (person,organization or geo-political) to an appropriate entry in a knowledge base (KB), which may or may not contain the entity. More formally, given a large corpus $\mathcal{C}$, let $q = (q.id, q.string, q.text)$ denote a query in the task which is a triple consisting of query id ($q.id$), name string ($q.string$) and context document ($q.text \in \mathcal{C}$). Let $o^{(q)} = \left\{o_1^{(q)}, \ldots, o_{n^{(q)}}^{(q)}\right\}$ denote the candidate KB entries associated with the query. Each KB entry is a tuple consisting of KB id, KB title, KB infobox (a set of attribute-value pairs that summarize or highlight the key features of the concept or subject of this entry) and KB text. The goal is to rank the KB entries and determine whether the top entry id should be considered as the answer, otherwise NIL should be returned.

A specific example of the task is as follows, given a name string "Michael Jordan" and its context document "...England Youth International goalkeeper *Michael Jordan*...". From the name string, we retrieve a set of candidate KB entries including "Michael Jordan (mycologist)", "Michael Jordan (footballer)", etc. The entity linking system should return the id of "Michael Jordan (footballer)" as the answer, rather than the id of "Michael Jordan" who is most well known as a basketball player.

4.2 General Framework

A general framework of entity linking consists of two crucial components, one for candidate generation, the other for candidate ranking, as shown in Figure 2. In this paper, we developed the first component by following the procedures described in (Chen et al., 2010) which extensively leveraged resources mined from Wikipedia. The performance of the first component is 96.8% measured by recall (the percentage of queries in which the candidates cover the true answer). We then focus on the second component.

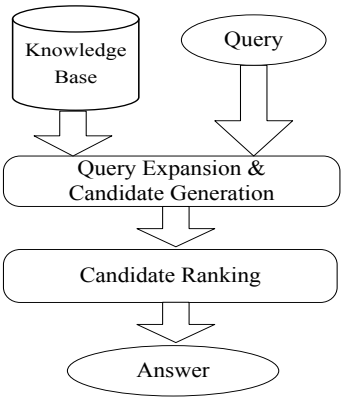


Figure 2: A general framework of entity linking system.

4.3 Baseline Rankers

We developed 8 baseline rankers, including 4 unsupervised rankers (f_1, f_2, f_3, f_4) and 4 supervised rankers(f_5, f_6, f_7, f_8).

•Naive (f_1): since the answer for each query can either be a KB id or NIL, the naive ranker simply outputs NIL for all queries.

•Entity (f_2): f_2 is defined as weighted combination of entity similarities in three types (person, organization and geo-political). Name entities are extracted from $q.text$ and KB text respectively using Stanford NER toolkit[3]. The formulas to compute entity similarities are defined in (Yoshida et al., 2010).

•Tfidf (f_3): f_3 is defined as cosine similarity between $q.text$ and KB text using tfidf weights.

•Profile (f_4): f_4 is defined as profile similarity between $q.text$ and KB text (Chen et al., 2010). We used a slot filling toolkit (Chen et al., 2011) to generate the profile (attribute-value pairs) for each query.

•Maxent (f_5): a pointwise ranker implemented using OpenNLP Maxent toolkit[4] which is based on maximum entropy model.

•SVM (f_6): a pointwise ranker implemented using SVM^{light} (Joachims, 1999).

•SVM ranking (f_7): a pairwise ranker implemented using SVM^{rank} (Joachims, 2006).

•ListNet (f_8): a listwise ranker presented in (Cao et al., 2007).

The four supervised rankers apply exactly the same set of features except that SVM ranking (f_7) needs to double expand the feature vector. The features are categorized into three levels, surface features (Dredze et al., 2010; Zheng et al., 2010), document features (Dredze et al., 2010; Zheng et al., 2010), and profiling features (entity slots that are extracted by the slot filling toolkit (Chen et al., 2011)).

4.4 MiCR for Entity Linking

We convert the collaborator searching problem into a clustering problem, i.e., for a given query q in the task, we retrieve at most $K = 300$ documents from the large corpus $\mathcal{C}$, each of which contains $q.string$; we then apply a clustering algorithm to generate clusters over the documents, and form query collaborators (excluding $q.text$) from the cluster that contains $q.text$.

We experimented the following two clustering approaches:

(1)*agglomerative clustering*: it iteratively merges clusters from singleton documents until a stop threshold is reached. Document similarity is defined as cosine similarity using tfidf weights. We applied group-average linking strategy to merge clusters (Manning et al., 2008).

(2)*graph-based clustering*: it iteratively partitions clusters from one single cluster until a stop threshold is reached. Document similarity is similarly defined as agglomerative clustering. We selected normalized spectral clustering as our clustering algorithm (Shi and Malik, 2000).

We first selected f_3 as our basic ranking function, and investigated whether the ranker can benefit from query collaborators formed by either agglomerative clustering or graph clustering. We implemented three versions of composite function g_1 (*max*, *min* and *average*), and experimented their performance using three unsupervised rankers f_2, f_3, f_4 respectively.

Last, we implemented three supervised versions of g_1 (Maxent, SVM and ListNet respectively) by adding cluster-level features and retraining the models in three supervised rankers f_5, f_6, f_8 respectively. Cluster-level features include maximum, minimum, average tfidf/entity similarities between the candidate and the query collaboration group.

4.5 MaCR for Entity Linking

We implemented two versions of composite function g_2, *average* and *voting*. Furthermore, we investigated how the performance can be affected by incrementally adding more rankers into the ranker set $\mathcal{F}^*$. To do so, we first sorted the 8 rankers according to their performance on the *development* set from the highest to the lowest, and starting with the highest performance ranker, we added one ranker at a time, until we have all the 8 rankers. It is worth noting that, when there are even number of rankers in the set $\mathcal{F}^*$, "ties" could take place using *voting* function. In order to break the ties, we rank the candidate higher if it is output as the answer from a higher performance ranker.

4.6 MiMaCR for Entity Linking

We investigated how the final performance can be boosted by jointly computing micro-ranking scores and macro-ranking score.

[3]http://nlp.stanford.edu/software/CRF-NER.shtml

[4]http://maxent.sourceforge.net/about.html

5 Experiments

5.1 Data and Evaluation Metric

We used TAC-KBP2009 evaluation data as our training (75%) and development set (25%), and used TAC-KBP2010 evaluation data as our blind testing set (shown in Table 2).

Corpus	Queries			
	PER	ORG	GPE	Total
Training&Dev	627	2710	567	3904
Testing	750	750	750	2250

Table 2: Training, development and testing corpus.

The reference KB consists of 818,741 entries which are extracted from an October 2008 dump of English Wikipedia. The source text corpus (denoted as $\mathcal{C}$ in section 4.1) consists of 1,777,888 documents in 5 genres (mostly Newswire and Web Text).

We used the official evaluation metric for TAC-KBP2010 entity linking task, that is, micro-averaged accuracy. It is computed by

$$\textit{micro-averaged accuracy} = \frac{\#\textit{correct answers}}{\#\textit{queries}}$$

An answer is considered as correct if the system output (either a KB entry id or NIL) exactly matches the key.

5.2 Performance of 8 Baseline Rankers

Table 3 shows the performance of the 8 baseline rankers in 4 columns: *Overall* for all queries, *PER* for person queries, *ORG* for organization queries, and *GPE* for geo-political queries. Each column is further split into *All*, *KB* (for Non-NIL queries) and *NIL* (for NIL queries). It shows that all the four supervised rankers perform better than the four unsupervised rankers. Naive ranker obtains the lowest overall micro-average accuracy (54.5%) but the highest NIL accuracy (100%). Among the four unsupervised rankers, *profile* ranker performs the best, which clearly shows that the extracted attributes of entities are effective for disambiguating confusable names. For example, our data analysis shows that the attribute value of "*per:alternative-name*" from the context document is particularly useful if a person query is only mentioned by its last name. The attribute "*per:title*" is another important indicator to discriminate one person from the other. For geo-political queries, if the query is a city name, attribute "*gpe:state*" is useful to distinguish cities with the same name but in different states or provinces. Among the four supervised rankers, ListNet outperforms SVM ranking and then SVM ranking outperforms the two pointwise rankers. It may confirm previous research findings that listwise ranking is superior to pairwise ranking and pairwise ranking is superior to pointwise ranking (Cao et al., 2007; Zheng et al., 2010). The best baseline ranker (ListNet) obtains an absolute overall accuracy gain of 26.6% over the naive ranker.

5.3 Impact of MiCR

To study the impact of MiCR, we first select f_3 (*tfidf* ranker) as our ranking function. Figure 3 shows the performance of applying different query collaborator searching strategies (graph or agglomerative clustering) and different versions of g_1 (average, max and min respectively). We intentionally adjust the meaning of threshold (x-axis) for both graph clustering and agglomerative clustering, such that at threshold 0, both clustering algorithms generate the largest number of clusters (i.e., each document is a cluster), and at threshold 1, they generate only one cluster. We now take the *average* function (Figure 3 (a)) into considerations, as graph clustering algorithm gradually partitions from one cluster (corresponding to threshold 1) to more clusters, the number of query collaborators gradually reduces, meanwhile, the accuracy gradually increases and reaches the highest (73.6%) at threshold of 0.45, which clearly shows that removing noisy collaborators in the query collaboration group can improve the performance. As the threshold continues dropping below 0.45, the number of query collaborators reduces and the performance significantly drops until it reaches the baseline performance of *tfidf* ranker (68.3%). It clearly shows that maintaining a controllable number of query collaborators can improve the performance. For the agglomerative clustering, it is the other story. As it continues merging from singleton clusters (corresponding to threshold 0) to one single cluster, the performance continues increasing until in the end it reaches the highest accuracy of 72.6%. However, unlike graph clustering, a peak never appears in the middle which implies that agglomerative clustering is inferior to graph clustering.

	Overall (%)			PER (%)			ORG (%)			GPE (%)		
	All	KB	NIL	All	KB	NIL	All	KB	NIL	All	KB	NIL
Naive	54.5	0.0	**100**	70.8	0.0	**100**	59.7	0	**100**	33.0	0	**100**
Entity	65.6	48.6	79.7	82.1	52.1	94.5	68.4	46.2	83.3	46.1	48.5	41.3
Tfidf	68.3	45.0	87.7	83.6	54.3	95.7	66.2	45.9	80.0	54.9	40.3	84.6
Profile	75.0	58.7	88.6	90.8	82.2	94.4	73.3	62.7	80.4	61.0	46.1	91.1
Maxent	77.4	72.3	81.6	86.5	**82.6**	94.4	73.3	62.7	80.4	61.0	**71.5**	72.1
SVM	78.1	**73.0**	82.3	91.1	81.7	94.9	78.7	**70.0**	84.6	64.4	71.1	51.0
SVM Rank	80.3	66.7	91.7	**91.3**	76.3	97.6	77.3	59.7	89.1	72.3	66.7	83.8
ListNet	**81.1**	69.7	90.6	90.8	77.6	96.2	**79.0**	64.0	89.1	**73.5**	69.7	81.4

Table 3: Comparison of 8 baseline rankers.

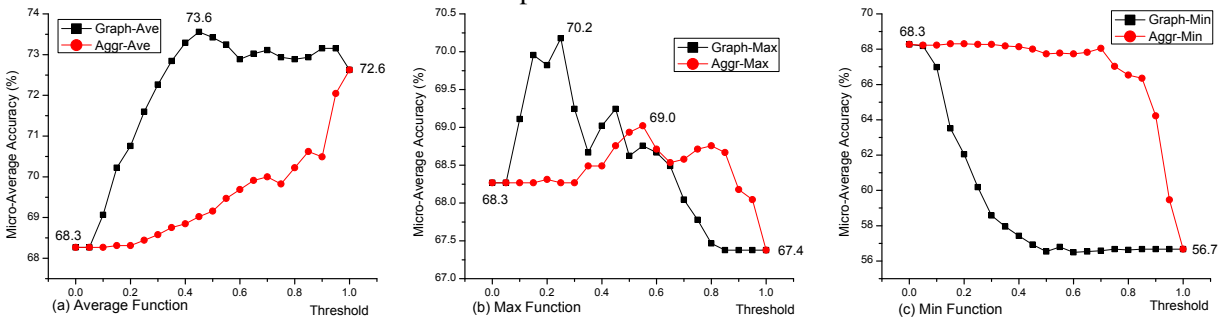


Figure 3: MiCR: comparison of average, max, and min functions combined with Graph and Agglomerative (Aggr)-based query collaborator searching strategies (*tfidf* ranker).

The *max* function (Figure 3 (b)) leverages the strengths from the strongest collaborator in the group, which can potentially improve KB accuracy, but meanwhile hurt NIL accuracy. As shown in the figure, as more collaborators join in the group, the performance increases first for both graph and agglomerative clustering, however, it starts to deteriorate when arriving at a threshold, and in the end, the performance drops even lower than the baseline of tfidf ranker.

The *min* function (Figure 3 (c)) leverages the strengths from the weakest collaborator in the group, which can potentially improve NIL accuracy, but meanwhile hurt KB accuracy. Our data analysis shows that the gain in NIL accuracy can not afford the larger loss in non-NIL accuracy, therefore, the performance continues dropping as the threshold increases. Min function is a counter example showing that searching query collaborators can not always lead to benefits.

To summarize so far, the best strategy for *tfidf* ranker in MiCR approach is *graph-ave* (applying graph clustering and using average function) which obtains overall accuracy gain of 5.3% over the baseline (68.3%). We further validate the performance of *graph-ave* using f_2, f_4 ranking functions, for *entity* ranker, we obtain accuracy gain of 6.3%, and for *profile* ranker, we obtain accuracy gain of 3.0%.

We then experiment the three supervised g_1 functions (ListNet, Maxent, and SVM respectively) using graph clustering as the query collaborator searching strategy. Figure 6 shows that ListNet, Maxent, SVM rankers obtain accuracy gain of 1.4%, 4.6%, 4.2% respectively over the baselines (corresponding to those points at threshold 0).

5.4 Impact of MaCR

Figure 4 shows that the MaCR approach obtains absolute accuracy gain of 1.3% (*voting* function) and 0.5% (*average* function) over the best baseline ranker (81.1%) when we add the 7th ranker (*entity* ranker). The improvement of *voting* function is statistically significant at a 99.6% confidence level by conducting Wilcoxon Matched-Pairs Signed-Ranks Test on the 10 folds of the testing set. However, the improvement of average function is not significant at the 0.05 level which implies that *average* is inferior to *voting*. We observe that the performance drops

when there are even number of rankers in the ranker set using voting function, which implies that our tie breaking strategy is not very effective.

We also experimented the *voting* function on the top 10 KBP2009 entity linking systems (each system performance is shown in the table embedded in Figure 5, and experiment is similarly done as described in section 4.5). Figure 5 shows that it can obtain absolute accuracy gain of 4.7% over the top entity linking system (82.2%). The reasons why we achieve relative smaller gains using our own ranker set are as follows: (1) we use the same candidate object set for all rankers, while different KBP2009 systems may use their own set of objects. (2) our top 4 supervised rankers apply almost the same set of features, while different KBP2009 systems may apply more diversified features. Therefore, diversity is a highly important factor that makes MaCR approach effective.

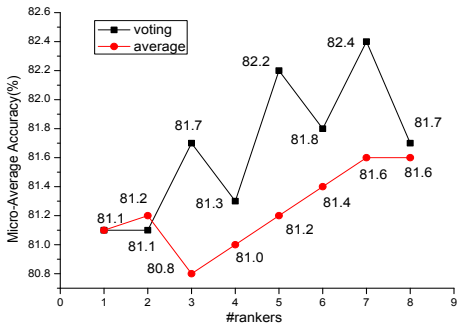


Figure 4: MaCR: comparison of voting and average.

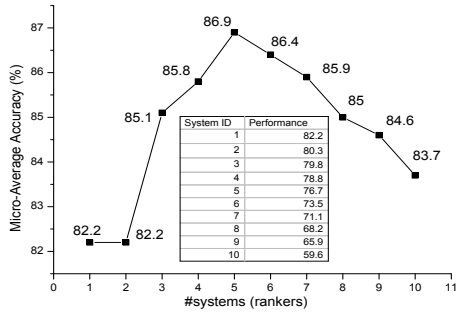


Figure 5: MaCR: applying voting function to the top 10 KBP2009 entity linking systems.

5.5 Impact of MiMaCR

We applied the following settings in our MiMaCR approach: selecting graph clustering as the query collaborator searching strategy, including five rankers (tfidf, entity, Maxent, SVM and ListNet) in the ranker set, using *average* function to compute micro-ranking scores for the tfidf and entity ranker, using the three corresponding supervised versions of g_1 to compute micro-ranking scores for Maxent, SVM and ListNet respectively, and finally applying *voting* function to compute the macro-ranking score. In Figure 6, the curve of "MiMaCR" shows how the performance of MiMaCR is affected by the threshold in graph clustering. We obtain the best micro-average accuracy of 83.7% at threshold 0.3, which is 2.6 % higher than the best baseline ranker (81.1%). The improvement is statistically significant at a 98.6% confidence level by conducting Wilcoxon Matched-Pairs Signed-Ranks Test on the 10 folds of the testing set. The score reported here is on par with the second best in the KBP2010 evaluation.

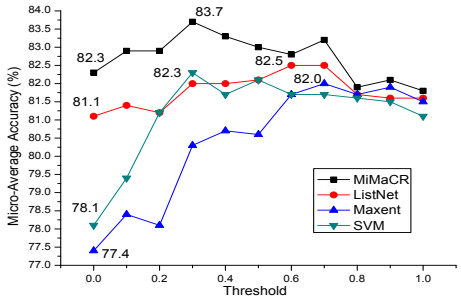


Figure 6: MiMaCR: Comparison of MiMaCR and three supervised versions of g_1 (ListNet, Maxent, and SVM respectively).

6 Related Work

In the literature of information retrieval, query expansion is a useful technique that involves the process of reformulating a query, and as a consequence, is capable to extend the ability of a query and improve the retrieval performance. Various approaches for query expansion have been proposed, as summarized in (Manning et al., 2008). The MiCR presented in this paper is superior to query expansion in two aspects, firstly, we leverage more information contained in multiple query collaborators; secondly, we place great emphasis on interactions among members in the query collaboration group.

In the literature of machine learning, there has been a considerable amount of research on ensemble-based classification, which is to build a predictive classification model by integrating multiple classifiers. A comprehensive survey is presented in (Rokach, 2009). In contrast, ensemble-based ranking has only recently attracted research interests (Hoi and Jin, 2008; Wei et al., 2010). Although the MaCR presented here is in essence ensemble-based

ranking, we extend it to MiMaCR which integrates the strengths from both MiCR and MaCR.

It is worth noting that "collaborative ranking" presented here should be distinguished from "collaborative filtering" in that "collaborative filtering" uses the known preferences of a group of users to generate personalized recommendations while "collaborative ranking" leverages query collaborators and ranker collaborators to enhance the overall ranking performance.

There has been an increasing amount of research on entity linking, especially through KBP2009 and KBP2010. Various unsupervised or supervised approaches have been proposed, as summarized in (McNamee and Dang, 2009; Ji et al., 2010). However, most of the previous research mainly focused on one or two ranking algorithms on isolated queries. In this paper, we have extended the work by systematically studying the possibility of performance enhancement through query-level collaboration and ranker-level collaboration.

7 Conclusions

We presented a new ranking scheme called *collaborative ranking* with three specific forms, MiCR, MaCR and MiMaCR and demonstrated its effectiveness on entity linking task. However, our scheme is not restricted to this specific task and it is generally applicable to many other other applications such as question answering. In MiCR, effective searching of query collaborators and active interplay among members in the query collaboration group are two key factors that make MiCR successful. In MaCR, diversity is a highly important factor to make it successful. Overall, MiMaCR can bootstrap the performance to its maximum if integrating MiCR and MaCR properly. However, the better performance is at the expense of much more computations.

Acknowledgments

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, the U.S. NSF CAREER Award under Grant IIS-0953149 and PSC-CUNY Research Program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

P. L. Bartlett, M. I. Jordan and J. D. McAuliffe. 2003. Convexity, classification, and risk bounds. *Technical Report 638*, Statistics Department, University of California, Berkeley.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22th International Conference on Machine Learning (ICML 2005)*.

Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai and H. Li. 2007. Learning to rank: from pairwise approach to listwise approach In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 129-136.

E. Charniak and M. Johnson. 2005. Coarseto-fine-grained n-best parsing and discriminative reranking. In *ACL-05*, pages 173-180.

W. Chen, T.-Y. Liu, Y. Lan, Z. Ma, and H. Li. 2009. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 315-323.

Z. Chen, S. Tamang, A. Lee, X. Li, W.-P. Lin, M. Snover, J. Artiles, M. Passantino and H. Ji. 2010. CUNYBLENDER TAC-KBP2010 Entity Linking and Slot Filling System Description. In *Proceedings of Text Analytics Conference (TAC2010)*.

Z. Chen, S. Tamang, A. Lee and H. Ji. 2011. A Toolkit for Knowledge Base Population. In *SIGIR*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 175-182.

M. Dredze, P. McNamee, D. Rao, A. Gerber and T. Finin. 2010. Entity Disambiguation for Knowledge Base Population. In *Proc. COLING 2010*.

Y. Freund, R. Iyer, R. Schapire, and Y. Singer. 2003. An efficient boosting algorithm for combining preferences. In *Journal of Machine Learning Research*, 4:933-969.

S. Hoi and R. Jin. 2008. Semi-supervised ensemble ranking. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence*.

L. Huang. 2008. Forest Reranking: Discriminative Parsing with Non-Local Features. In *ACL-HLT-08*, pages 586-594.

H. Ji, R. Grishman, H. T. Dang and K. Griffit. 2010. An Overview of the TAC2010 Knowledge Base Population Track. In *Proceedings of Text Analytics Conference (TAC2010)*.

T. Joachims. 1999. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD 2002)*.

T. Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Y. Lan, T.-Y. Liu, T. Qin, Z. Ma, and H. Li. 2008. Query-level stability and generalization in learning to rank. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 512-519.

P. Li, C. Burges, and Q. Wu. 2007. Mcrank: Learning to rank using multiple classification and gradient boosting In *Advances in Neural Information Processing Systems 20 (NIPS2007)*.

Y. Lin. 2002. Support vector machines and the bayes rule in classification. In *Data Mining and Knowledge Discovery*, pages 259-275.

C. D. Manning, P. Raghavan and H. Schütze. 2008 . Introduction to Information Retrieval. Cambridge University Press.

P. McNamee and H. Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. In *Proceedings of TAC*.

R. Nallapati. 2004. Discriminative models for information retrieval. In *SIGIR*.

F. J. Och. 2002. Statistical Machine Translation: From Single-Word Models to Alignment Templates. *Ph.D. thesis*, Computer Science Department, RWTH Aachen, Germany, October.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. *Technical report*, Stanford Digital Library Technologies Project.

T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu and H. Li. 2007. Query-level loss functions for information retrieval. In *Information Processing and Management*.

T. Qin, T.-Y. Liu, X.-D. Zhang,D.-S. Wang, and H. Li. 2008. Global Ranking Using Continuous Conditional Random Fields. In *Advances in Neural Information Processing Systems 21 (NIPS 2008)*.

D. Ravichandran, E. Hovy and F. J. Och. 2003. Statistical QA - Classifier vs. Re-ranker: What's the difference? In *Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering*.

S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal and Y. Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of ACL*.

L. Rokach. 2009. Ensemble-based classifiers. Artif Intell Rev DOI 10.1007/s10462-009-9124-7.

L. Shen, A. Sarkar, and F. J. Och. 2005. Discriminative reranking for machine translation. In *Proceedings of HLT-NAACL*.

J. Shi and J. Malik. 2000. Normalized Cuts and Image Segmentation. In *Machine Intelligence*, vol. 22, no. 8, pages 888-905.

F. Wei, W. Li and S. Liu. 2010. iRANK: A rank-learn-combine framework for unsupervised ensemble ranking. In *Journal of the American Society for Information Science and Technology*,61: 1232C1243. doi: 10.1002/asi.21296.

X. Yang and J. Su and C.L. Tan 2008. A Twin-Candidate Model for Learning-based Anaphora Resolution. In *Computational Linguistics*, vol. 34, no. 3, pages 327-356.

M. Yoshida, M. Ikeda, S. Ono, I. Sato, and H. Nakagawa. 2010. Person name disambiguation by boostrapping. In *SIGIR*.

T. Zhang. 2004. Statistical analysis of some multicategory large margin classification methods. In *Journal of Machine Learning Research*, 5, 1225-1251.

W. Zhang, J. Su, C. L. Tan and W.T. Wang. 2010. Entity Linking Leveraging Automatically Generated Annotation. In *Proc. COLING 2010*.

Z. Zheng, F. Li, M. Huang, X. Zhu. 2010. Learning to Link Entities with Knowledge Base. In *Proc. HLT-NAACL2010*.

Robust Disambiguation of Named Entities in Text

Johannes Hoffart[1], **Mohamed Amir Yosef**[1], **Ilaria Bordino**[2], **Hagen Fürstenau**[3], **Manfred Pinkal**[3], **Marc Spaniol**[1], **Bilyana Taneva**[1], **Stefan Thater**[3], **Gerhard Weikum**[1]
1 Max Planck Institute for Informatics, Saarbrücken, Germany
2 Yahoo! Research Lab, Barcelona, Spain
3 Saarland University, Saarbrücken, Germany
{jhoffart,mamir,mspaniol,btaneva,weikum}@mpi-inf.mpg.de
bordino@yahoo-inc.com {hagenf,pinkal,stth}@coli.uni-sb.de

Abstract

Disambiguating named entities in natural-language text maps mentions of ambiguous names onto canonical entities like people or places, registered in a knowledge base such as DBpedia or YAGO. This paper presents a robust method for collective disambiguation, by harnessing context from knowledge bases and using a new form of coherence graph. It unifies prior approaches into a comprehensive framework that combines three measures: the prior probability of an entity being mentioned, the similarity between the contexts of a mention and a candidate entity, as well as the coherence among candidate entities for all mentions together. The method builds a weighted graph of mentions and candidate entities, and computes a dense subgraph that approximates the best joint mention-entity mapping. Experiments show that the new method significantly outperforms prior methods in terms of accuracy, with robust behavior across a variety of inputs.

1 Introduction

1.1 Motivation

Web pages, news articles, blog postings, and other Internet data contain mentions of named entities such as people, places, organizations, etc. Names are often ambiguous: the same name can have many different meanings. For example, given a text like "They performed Kashmir, written by Page and Plant. Page played unusual chords on his Gibson.", how can we tell that "Kashmir" denotes a song by Led Zeppelin and not the Himalaya region (and that Page refers to guitarist Jimmy Page and not to Google founder Larry Page, and that Gibson is a guitar model rather than the actor Mel Gibson)?

Establishing these mappings between the mentions and the actual entities is the problem of *named-entity disambiguation (NED)*.

If the possible meanings of a name are known upfront - e.g., by using comprehensive gazetteers such as GeoNames (`www.geonames.org`) or knowledge bases such as DBpedia (Auer07), Freebase (`www.freebase.com`), or YAGO (Suchanek07), which have harvested Wikipedia redirects and disambiguation pages - then the simplest heuristics for name resolution is to choose the most prominent entity for a given name. This could be the entity with the longest Wikipedia article or the largest number of incoming links in Wikipedia; or the place with the most inhabitants (for cities) or largest area, etc. Alternatively, one could choose the entity that uses the mention most frequently as a hyperlink anchor text. For the example sentence given above, all these techniques would incorrectly map the mention *"Kashmir"* to the Himalaya region. We refer to this suite of methods as a *popularity-based (mention-entity) prior*.

Key to improving the above approaches is to consider the *context* of the mention to be mapped, and compare it - by some *similarity measure* - to contextual information about the potential target entities. For the example sentence, the mention "Kashmir" has context words like "performed" and "chords" so that we can compare a bag-of-words model against characteristic words in the Wikipedia articles of the different candidate entities (by measures such as cosine similarity, weighted Jaccard distance, KL divergence, etc.). The candidate entity with the highest similarity is chosen. Alternatively, labeled training data can be harnessed to learn a multi-way classifier, and additional features like entire phrases, part-of-speech tags, dependency-parsing paths, or nearby

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 782–792,
Edinburgh, Scotland, UK, July 27–31, 2011.

hyperlinks can be leveraged as well. These methods work well for sufficiently long and relatively clean input texts such as predicting the link target of a Wikipedia anchor text (Milne08). However, for short or more demanding inputs like news, blogs, or arbitrary Web pages, relying solely on context similarity cannot achieve near-human quality. Similarity measures based on syntactically-informed distributional models require minimal context only. They have been developed for common nouns and verbs (Thater10), but not applied to named entities.

The key to further improvements is to jointly consider multiple mentions in an input and aim for a *collective assignment* onto entities (Kulkarni09). This approach should consider the *coherence* of the resulting entities, in the sense of semantic relatedness, and it should combine such measures with the context similarity scores of each mention-entity pair. In our example, one should treat "Page", "Plant" and "Gibson" also as named-entity mentions and aim to disambiguate them together with "Kashmir".

Collective disambiguation works very well when a text contains mentions of a sufficiently large number of entities within a thematically homogeneous context. If the text is very short or is about multiple, unrelated or weakly related topics, collective mapping tends to produce errors by directing some mentions towards entities that fit into a single coherent topic but do not capture the given text. For example, a text about a football game between "Manchester" and "Barcelona" that takes place in "Madrid" may end up mapping either all three of these mentions onto football clubs (i.e., Manchester United, FC Barcelona, Real Madrid) or all three of them onto cities. The conclusion here is that none of the prior methods for named-entity disambiguation is robust enough to cope with such difficult inputs.

1.2 Contribution

Our approach leverages recently developed knowledge bases like YAGO as an entity catalog and a rich source of entity types and semantic relationships among entities. These are factored into new measures for the similarity and coherence parts of collectively disambiguating all mentions in an input text. For similarity, we also explore an approach that leverages co-occurrence information obtained from large, syntactically parsed corpora (Thater10).

We cast the joint mapping into the following graph problem: mentions from the input text and candidate entities define the node set, and we consider weighted edges between mentions and entities, capturing context similarities, and weighted edges among entities, capturing coherence. The goal on this combined graph is to identify a dense subgraph that contains exactly one mention-entity edge for each mention, yielding the most likely disambiguation. Such graph problems are NP-hard, as they generalize the well-studied Steiner-tree problem. We develop a greedy algorithm that provides high-quality approximations, and is customized to the properties of our mention-entity graph model.

In addition to improving the above assets for the overall disambiguation task, our approach gains in robustness by using components selectively in a self-adapting manner. To this end, we have devised the following multi-stage procedure.

- For each mention, we compute popularity priors and context similarities for all entity candidates as input for our tests.
- We use a threshold test on the prior to decide whether popularity should be used (for mentions with a very high prior) or disregarded (for mentions with several reasonable candidates).
- When both the entity priors and the context similarities are reasonably similar in distribution for all the entity candidates, we keep the best candidate and remove all others, fixing this mention *before* running the coherence graph algorithm.

We then run the coherence graph algorithm on all the mentions and their remaining entity candidates. This way, we restrict the coherence graph algorithm to the critical mentions, in situations where the goal of coherence may be misleading or would entail high risk of degradation.

The paper makes the following novel contributions: 1) a framework for combining popularity priors, similarity measures, and coherence into a robust disambiguation method; 2) new measures for defining mention-entity similarity; 3) a new algorithm for computing dense subgraphs in a mention-entity graph, which produces high-quality mention-entity mappings; 4) an empirical evaluation on a demanding corpus (based on additional annotations for the dataset of the CoNLL 2003 NER task), with signifi-

cant improvements over state-of-the-art opponents.

2 State of the Art

Recognizing named entities (NER tagging) in natural-language text has been extensively addressed in NLP research. The output is labeled noun phrases. However, these are not yet canonical entities, explicitly and uniquely denoted in a knowledge repository. Approaches that use Wikipedia for explicit disambiguation date back to (Bunescu06) and have been further pursued by (Cucerzan07; Han09; Milne08; Nguyen08; Mihalcea07). (Bunescu06) defined a similarity measure that compared the context of a mention to the Wikipedia categories of an entity candidate. (Cucerzan07; Milne08; Nguyen08) extended this framework by using richer features for the similarity comparison. (Milne08) additionally introduced a supervised classifier for mapping mentions to entities, with learned feature weights rather than using the similarity function directly. (Milne08) introduced a notion of semantic relatedness between a mention's candidate entities and the unambiguous mentions in the textual context. The relatedness values are derived from the overlap of incoming links in Wikipedia articles. (Han09) considered another feature: the relatedness of common noun phrases in a mention's context, matched against Wikipedia article names. While these features point towards semantic coherence, the approaches are still limited to mapping each mention separately. Nonetheless, this line of feature-rich similarity-driven methods achieved very good results in experiments, especially for the task of predicting Wikipedia link targets for a given href anchor text. On broader input classes such as news articles (called "wikification in the wild" in (Milne08)), the precision was reported to be about 75 percent.

The first work with an explicit collective-learning model for joint mapping of all mentions has been (Kulkarni09). This method starts with a supervised learner for a similarity prior, and models the pairwise coherence of entity candidates for two different mentions as a probabilistic factor graph with all pairs as factors. The MAP (maximum a posteriori) estimator for the joint probability distribution of all mappings is shown to be an NP-hard optimization problem, so that (Kulkarni09) resorts to approximations and heuristics like relaxing an integer linear program (ILP) into an LP with subsequent rounding or hill-climbing techniques. The experiments in (Kulkarni09) show that this method is superior to the best prior approaches, most notably (Milne08). However, even approximate solving of the optimization model has high computational costs.

Coreference resolution is the task of mapping mentions like pronouns or short phrases to a preceding, more explicit, mention. Recently, interest has arisen in cross-document coreference resolution (Mayfield09), which comes closer to NED, but does not aim at mapping names onto entities in a knowledge base. Word sense disambiguation (McCarthy09; Navigli09) is the more general task of mapping content words to a predefined inventory of word senses. While the NED problem is similar, it faces the challenges that the ambiguity of entity names tends to be much higher (e.g., mentions of common lastnames or firstname-only).

Projects on automatically building knowledge bases (Doan08) from natural-language text include KnowItAll (Banko07), YAGO and its tool SOFIE (Suchanek09; Nakashole11), StatSnowball (Zhu09), ReadTheWeb (Carlson10), and the factor-graph work by (Wick09). Only SOFIE maps names onto canonical entities; the other projects produce output with ambiguous names. SOFIE folds the NED into its MaxSat-based reasoning for fact extraction. This approach is computationally expensive and not intended for online disambiguation of entire texts.

3 Framework

Mentions and Ambiguity: We consider an input text (Web page, news article, blog posting, etc.) with mentions (i.e., surface forms) of named entities (people, music bands, songs, universities, etc.) and aim to map them to their proper entries in a knowledge base, thus giving a disambiguated meaning to entity mentions in the text. We first identify noun phrases that potentially denote named entities. We use the Stanford NER Tagger (Finkel05) to discover these and segment the text accordingly.

Entity Candidates: For possible entities (with unique canonical names) that a mention could denote, we harness existing knowledge bases like DBpedia or YAGO. For each entity they provide a set of short names (e.g., "Apple" for `Apple Inc.` and para-

phrases (e.g., "Big Apple" for `New York City`). In YAGO, these are available by the `means` relation, which in turn is harvested from Wikipedia disambiguation pages, redirects, and links.

Popularity Prior for Entities: Prominence or popularity of entities can be seen as a probabilistic prior for mapping a name to an entity. The most common way of estimating this are the Wikipedia-based frequencies of particular names in link anchor texts referring to specific entities, or number of inlinks.

Context Similarity of Mentions and Entities: The key for mapping mentions onto entities are the contexts on both sides of the mapping. We consider two different approaches. First, for each mention, we construct a context from all words in the entire input text. This way, we can represent a mention as a set of (weighted) words or phrases that it co-occurs with. Second, we alternatively consider similarity scores based on syntactically-parsed contexts, based on (Thater10). On the entity side of the mapping, we associate each entity with characteristic keyphrases or salient words, precomputed from Wikipedia articles and similar sources. For example, `Larry Page` would have keyphrases like "Stanford", "search engine", etc., whereas `Jimmy Page` may have keyphrases "Gibson guitar", "hard rock", etc. Now we can define and compute *similarity* measures between a mention and an entity candidate, e.g., the weighted word overlap, the KL divergence, n-gram-based measures, etc. In addition, we may use syntactic contextualization techniques, based on dependency trees, that suggest phrases that are typically used with the same verb that appears with the mention in the input text (Thater10).

Coherence among Entities: On the entity side, each entity has a context in the underlying knowledge base(s): other entities that are connected via semantic relationships (e.g., `memberOf`) or have the same semantic type (e.g., `rock musician`). An asset that knowledge bases like DBpedia and YAGO provide us with is the same-as cross-referencing to Wikipedia. This way, we can quantify the coherence between two entities by the number of incoming links that their Wikipedia articles share. When we consider candidate entities for different mentions, we can now define and compute a notion of *coherence* among the corresponding entities, e.g., by the overlap among their related entities or some form of type distance. Coherence is a key asset because most texts deal with a single or a few semantically related topics such as rock music or Internet technology or global warming, but not everything together.

Overall Objective Function: To aim for the best disambiguation mappings, our framework combines prior, similarity, and coherence measures into a combined objective function: for each mention m_i, $i = 1..k$, select entity candidates e_{j_i}, one per mention, such that

$$\alpha \cdot \sum_{i=1..k} \text{prior}(m_i, e_{j_i}) +$$
$$\beta \cdot \sum_{i=1..k} \text{sim}(\text{cxt}(m_i), \text{cxt}(e_{j_i})) +$$
$$\gamma \cdot \text{coh}(e_{j_1} \in \text{cnd}(m_1) \ldots e_{j_k} \in \text{cnd}(m_k)) = \max!$$

where $\alpha + \beta + \gamma = 1$, $\text{cnd}(m_i)$ is the set of possible meanings of m_i, cxt( ) denotes the context of mentions and entities, respectively, and coh( ) is the coherence function for a set of entities.

Section 4 gives details on each of these three components. For robustness, our solution selectively enables or disables the three components, based on tests on the mentions of the input text; see Section 5.

4 Features and Measures

4.1 Popularity Prior

As mentioned above, our framework supports multiple forms of popularity-based priors, but we found a model based on Wikipedia link anchors to be most effective: For each surface form that constitutes an anchor text, we count how often it refers to a particular entity. For each name, these counts provide us with an estimate for a probability distribution over candidate entities. For example, "Kashmir" refers to `Kashmir` (the region) in 90.91% of all occurrences and in 5.45% to `Kashmir (Song)`.

4.2 Mention-Entity Similarity

Keyphrase-based Similarity: On the **mention side**, we use all tokens in the document (except stopwords and the mention itself) as context. We experimented with a distance discount to discount the weight of tokens that are further away, but this did not improve the results for our test data.

On the **entity side**, the knowledge base knows authoritative sources for each entity, for example, the

corresponding Wikipedia article or an organizational or individual homepage. These are the inputs for an offline data-mining step to determine characteristic *keyphrases* for each entity and their statistical weights. We describe this only for Wikipedia as input corpus, the approach extends to other inputs. As keyphrase candidates for an entity we consider its corresponding Wikipedia article's link anchors texts, including category names, citation titles, and external references. We extended this further by considering also the titles of articles linking to the entity's article. All these phrases form the keyphrase set of an entity: $KP(e)$.

For each word w that occurs in a keyphrase, we compute a *specificity weight* with regard to the given entity: the **MI** (mutual information) between the entity e and the keyword w, calculating the joint probabilities for MI as follows:

$$p(e,w) = \frac{\left|w \in \left(KP(e) \cup \bigcup_{e' \in IN_e} KP(e')\right)\right|}{N}$$

reflecting if w is contained in the keyphrase set of e or any of the keyphrase sets of an entity linking to e, $IN(e)$, with N denoting the total number of entities. The joint probabilities for the cases $p(e,\bar{w})$, $p(\bar{e},w)$, $p(\bar{e},\bar{w})$ are calculated accordingly.

Keyphrases may occur only partially in an input text. For example, the phrase "Grammy Award winner" associated with entity `Jimmy Page` may occur only in the form "Grammy winner" near some mention "Page". Therefore, our algorithm for the similarity of mention m with regard to entity e computes partial matches of e's keyphrases in the text. This is done by matching individual words and rewarding their proximity in an appropriate score. To this end we compute, for each keyphrase, the shortest window of words that contains a maximal number of words of the keyphrase. We refer to this window as the phrase's *cover* (cf. (Taneva11)). For example, matching the text "winner of many prizes including the Grammy" results in a cover length of 7 for the keyphrase "Grammy award winner". By this rationale, the score of partially matching phrase q in a text is set to:

$$score(q) = z \left(\frac{\sum_{w \in cover} weight(w)}{\sum_{w \in q} weight(w)} \right)^2$$

where $z = \frac{\#\ matching\ words}{length\ of\ cover(q)}$ and $weight(w)$ is either the **MI** weight (defined above) or the collection-wide **IDF** weight of the keyphrase word w. Note that the second factor is squared, so that there is a superlinear reduction of the score for each word that is missing in the cover.

For the similarity of a mention m to candidate entity e, this score is aggregated over all keyphrases of e and all their partial matches in the text, leading to the *similarity score*

$$simscore(m,e) = \sum_{q \in KP(e)} score(q)$$

Syntax-based Similarity: In addition to surface features of words and phrases, we leverage information about the immediate syntactic context in which an entity mention occurs. For example, in the sentence "Page played unusual chords", we can extract the fact that the mention "Page" is the subject of the verb "play". Using a large text corpus for training, we collect statistics about what kinds of entities tend to occur as subjects of "play", and then rank the candidate entities according to their compatibility with the verb.

Specifically, we employ the framework of (Thater10), which allows us to derive vector representations of words in syntactic contexts (such as being the subject of a particular verb). We do not directly apply this model to derive contextualized representations of entity mentions, as information about specific proper names is very sparse in corpora like GigaWord or Wikipedia. Instead, we consider a set of *substitutes* for each possible entity e, which we take as its context $\text{cxt}(e)$. For this, we use the WordNet synsets associated with the entity's YAGO types and all their hypernyms. For each substitute, we compute a standard distributional vector and a contextualized vector according to (Thater10). Syntax-based similarity between $\text{cxt}(e)$ and the context $\text{cxt}(m)$ of the mention is then defined as the sum of the scalar-product similarity between these two vectors for each substitute. This results in high similarity if the syntactic contextualization only leads to small changes of the vectors, reflecting the compatibility of the entity's substitutes.

In our example, we compute a vector for "guitarist" as subject of "play", and another one for "entrepreneur" in the same context. The former is more

compatible with the given context than the latter, leading to higher similarity for the entity `Jimmy Page`.

4.3 Entity-Entity Coherence

As all entities of interest are registered in a knowledge base (like YAGO), we can utilize the *semantic type system*, which is usually a DAG of classes. The simples measure is the distance between two entities in terms of `type` and `subclassOf` edges.

The knowledge bases also provide same-as cross-referencing to Wikipedia, amd we quantify the coherence between two entities by the number of *incoming links* that their Wikipedia articles share. This approach has been refined by Milne and Witten (Milne08), taking into account the total number N of entities in the (Wikipedia) collection:

$$mw\_coh(e_1, e_2) = 1 - \frac{\log\left(\max(|IN_{e_1}|, |IN_{e_2}|)\right) - \log(|IN_{e_1} \cap IN_{e_2}|)}{\log(|N|) - \log\left(\min(|IN_{e_1}|, |IN_{e_2}|)\right)}$$

if > 0 and else set to 0.

5 Graph Model and Algorithms

5.1 Mention-Entity Graph

From the popularity, similarity, and coherence measures discussed in Section 4, we construct a weighted, undirected graph with mentions and candidate entities as nodes. As shown in the example of Figure 1, the graph has two kinds of edges:

- A mention-entity edge is weighted with a similarity measure or a combination of popularity and similarity measure. Our experiments will use a linear combination with coefficients learned from withheld training data.
- An entity-entity edge is weighted based on Wikipedia-link overlap, or type distance, or some combination along these lines.

Our experiments will focus on anchor-based popularity, keyphrase-based and/or syntactic similarity, and link-based coherence (mw_coh). The mention-entity graph is dense on the entities side and often has hundreds or thousands of nodes, as the YAGO knowledge base offers many candidate entities for common mentions (e.g., country names that could also denote sports teams, common lastnames, firstnames, etc.).

5.2 Graph Algorithm

Given a mention-entity graph, our goal is to compute a *dense subgraph* that would ideally contain all mention nodes and exactly one mention-entity edge for each mention, thus disambiguating all mentions. We face two main challenges here. The first is how to specify a notion of density that is best suited for capturing the coherence of the resulting entity nodes. The seemingly most natural approach would be to measure the density of a subgraph in terms of its total edge weight. Unfortunately, this will not work robustly for the disambiguation problem. The solution could be dominated by a few entity nodes with very high weights of incident edges, so the approach could work for prominent targets, but it would not achieve high accuracy also for the long tail of less prominent and more sparsely connected entities. We need to capture the weak links in the collective entity set of the desired subgraph. For this purpose, we define the *weighted degree* of a node in the graph to be the total weight of its incident edges. We then define the density of a subgraph to be equal to the minimum weighted degree among its nodes. Our goal is to compute a subgraph with maximum density, while observing constraints on the subgraph structure.

The second critical challenge that we need to face is the computational complexity. Dense-subgraph problems are almost inevitably NP-hard as they generalize the Steiner-tree problem. Hence, exact algorithms on large input graphs are infeasible.

To address this problem, we adopt and extend an approximation algorithm of (Sozio10) for the problem of finding strongly interconnected, size-limited groups in social networks. The algorithm starts from the full mention-entity graph and iteratively removes the entity node with the smallest weighted degree. Among the subgraphs obtained in the various steps, the one maximizing the minimum weighted degree will be returned as output. To guarantee that we arrive at a coherent mention-entity mapping for all mentions, we enforce each mention node to remain connected to at least one entity. However, this constraint may lead to very suboptimal results.

For this reason, we apply a pre-processing phase to prune the entities that are only remotely related to the mention nodes. For each entity node, we compute the distance from the set of all mention nodes in terms

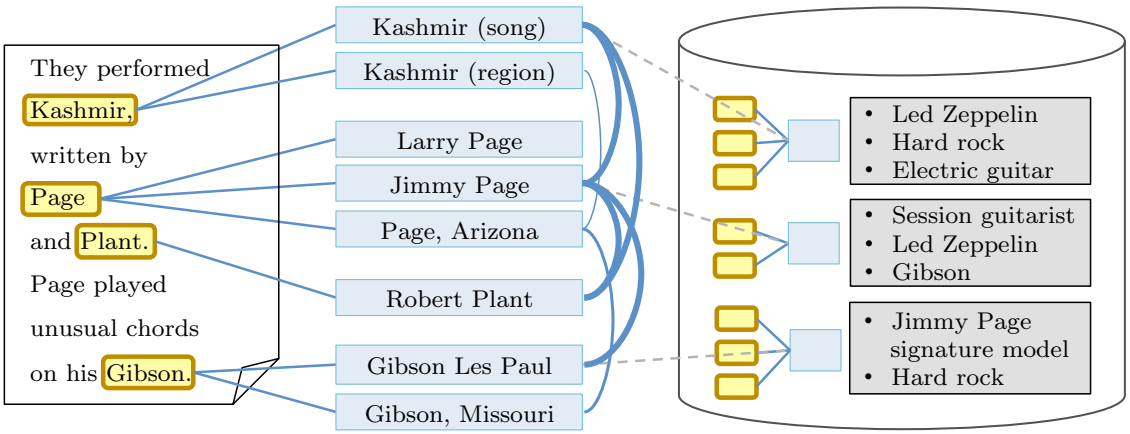


Figure 1: Mention-Entity Graph Example

of the sum of the corresponding squared shortest-path distances. We then restrict the input graph to the entity nodes that are closest to the mentions. An experimentally determined good choice for the size of this set is five times the number of the mention nodes. Then the iterative greedy method is run on this smaller subgraph. Algorithm 1 summarizes this procedure, where an `entity` is `taboo` if it is the last candidate for a mention it is connected to.

Algorithm 1: Graph Disambiguation Algorithm

Input: weighted graph of mentions and entities
Output: result graph with one edge per mention
begin
  pre–processing phase;
  foreach `entity` **do**
    calculate distance to all mentions;
  keep the closest ($5\times$ `mentions_count`) `entities`, drop the others;
  main loop;
  while *graph has* `non-taboo entity` **do**
    determine `non-taboo entity` node with lowest weighted degree, remove it and all its incident edges;
    if *minimum weighted degree increased* **then**
      set `solution` to current graph;
  post–processing phase;
  process `solution` by local search or full enumeration for best configuration;

The output of the main loop would often be close to the desired result, but may still have more than one mention-entity edge for one or more mentions. At this point, however, the subgraph is small enough to consider an exhaustive enumeration and assessment of all possible solutions. This is one of the options that we have implemented as post-processing step. Alternatively, we can perform a faster local-search algorithm. Candidate entities are randomly selected with probabilities proportional to their weighted degrees. This step is repeated for a prespecified number of iterations, and the best configuration with the highest total edge-weight is used as final solution.

5.3 Robustness Tests

The graph algorithm generally performs well. However, it may be misled in specific situations, namely, if the input text is very short, or if it is thematically heterogeneous. To overcome these problems, we introduce two *robustness tests* for individual mentions and, depending on the tests' outcomes, use only a subset of our framework's features and techniques.

Prior test: Our first test ensures that the popularity prior does not unduly dominate the outcome if the true entities are dominated by false alternatives. We check, for each mention, whether the popularity prior for the most likely candidate entity is above some threshold ρ, e. g. above 90% probability. If this is not the case, then the prior is completely disregarded for computing the mention-entity edge weights. Otherwise, the prior is combined with the context-based similarity computation to determine edge weights.

We never rely solely on the prior.

Coherence test: As a test for whether the coherence part of our framework makes sense or not, we compare the popularity prior and the similarity-only measure, on a per-mention basis. For each mention, we compute the $L1$ distance between the popularity-based vector of candidate probabilities and the similarity-only-based vector of candidate probabilities:

$$\sum_{i=1..k} |\text{prior}(m, e_i) - \text{simscore}(m, e_i)|$$

This difference is always between 0 and 2. If it exceeds a specified threshold λ (e.g., 1), the disagreement between popularity and similarity-only indicates that there is a situation that coherence may be able to fix. If, on the other hand, there is hardly any disagreement, using coherence as an additional aspect would be risky for thematically heterogeneous texts and should better be disabled. In that case, we choose an entity for the mention at hand, using the combination of prior and similarity. Only the winning entity is included in the mention-entity graph, all other candidates are omitted for the graph algorithm. The robustness tests and the resulting adaptation of our method are fully automated.

6 Experiments

6.1 Setup

System: All described methods are implemented in a prototype system called AIDA (Accurate Online Disambiguation of Named Entities). We use the Stanford NER tagger (Finkel05) to identify mentions in input texts, the YAGO2 knowledge base (Hoffart11) as a repository of entities, and the English Wikipedia edition (as of 2010-08-17) as a source of mining keyphrases and various forms of weights. The graph algorithm makes use of Webgraph (Boldi04).

Datasets: There is no established benchmark for NED. The best prior work (Kulkarni09)) compiled its own hand-annotated dataset, sampled from online news. Unfortunately, this data set is fairly small (102 short news articles, about 3,500 proper noun mentions). Moreover, its entity annotations refer to an old version of Wikipedia. To avoid unfair comparisons, we created our own dataset based on CoNLL 2003 data, extensively used in prior work on NER tagging (Sang03).

articles	1,393
mentions (total)	34,956
mentions with no entity	7,136
words per article (avg.)	216
mentions per article (avg.)	25
distinct mentions per article (avg.)	17
mentions with candidate in KB (avg.)	21
entities per mention (avg)	73
initial annotator disagreement (%)	21.1

Table 1: *CoNLL* Dataset Properties

This consists of proper noun annotations for 1393 Reuters newswire articles. We hand-annotated all these proper nouns with corresponding entities in YAGO2. Each mention was disambiguated by two students and resolved by us in case of conflict. This data set is referred to as *CoNLL* in the following and fully available at `http://www.mpi-inf.mpg.de/yago-naga/aida/`. Table 1 summarizes properties of the dataset.

Methods under comparison: Our framework includes many variants of prior methods from the literature. We report experimental results for some of them. AIDA's parameters were tuned by line-search on 216 withheld development documents. We found the following to work best:

- threshold for prior test: $\rho = 0.9$
- weights for popularity, similarity, coherence: $\alpha = 0.43$, $\beta = 0.47$, $\gamma = 0.10$
- initial number of entites in graph: $5 \cdot \#mentions$
- threshold for coherence test: $\lambda = 0.9$

We checked the sensitivity of the hyper-parameter settings and found the influence of variations to be small, e. g. when varying λ within the range [0.5,1.3], the changes in precision@1.0 are within 1%.

The baseline for our experiments is the collective-inference method of (Kulkarni09), which outperforms simpler methods (such as (Milne08)). We refer to this method as *Kul_CI*. Since program code for this method is not available, we re-implemented it using the LP solver CPLEX for the optimization problem with subsequent rounding, as described in (Kulkarni09). In addition, we compare against (our re-implementation of) the method of (Cucerzan07),

	Our Methods							Competitors				
	sim-k	prior sim-k	prior sim-s	sim-k sim-s	r-prior sim-k	r-prior sim-k coh	r-prior sim-k r-coh	prior	Cuc	Kul_s	Kul_sp	Kul_CI
Macro P@1.0	76.53	75.75	71.43	76.40	80.71	80.73	**81.91**	71.24	43.74	58.06	76.74	76.74
Micro P@1.0	76.09	70.72	66.09	76.13	79.57	81.77	**81.82**	65.84	51.03	63.42	72.31	72.87
MAP	66.98	83.99	85.97	67.00	85.91	**89.05**	87.31	86.63	40.06	63.90	86.50	85.44

Table 2: Experimental results on *CoNLL* (all values in %)

referred to as *Cuc*. For all methods, weights for combining components were obtained by training a SVM classifier on 946 withheld *CoNLL* training documents.

Performance measures: The key measures in our evaluation are *precision* and *recall*. We consider the precision-recall curve, as there is an inherent trade-off between the two measures. Precision is the fraction of mention-entity assignments that match the ground-truth assignment. Recall is the fraction of the ground-truth assignments that our method(s) could compute. Both measures can aggregate over of all mentions (across all texts) or over all input texts (each with several mentions). The former is called micro-averaging, the latter macro-averaging.

As we use a knowledge base with millions of entities, we decided to neglect the situation that a mention may refer to an unknown entity not registered in the knowledge base. We consider only mention-entity pairs where the ground-truth gives a known entity, and thus ignore roughly 20% of the mentions without known entity in the ground-truth. This simplifies the calculation of aggregated precision-recall measures like (interpolated) MAP (mean average precision):

$$\text{MAP} = \frac{1}{m} \sum_{i=1..m} \text{precision@}\frac{i}{m}$$

where precision@$\frac{i}{m}$ is the precision at a specific recall level. This measure is equivalent to the area under the precision-recall curve.

For constructing the precision-recall curve, we sort the mention-entity pairs in descending order of confidence, so that x% recall refers to the x% with the highest confidence. We use each method's mention-entity similarity for the confidence values.

6.2 Results

The results of AIDA vs. the collective-inference method of (Kulkarni09) and the entity disambiguation method of (Cucerzan07) on 229 test documents are shown in Table 2[1]. The table includes variants of our framework, with different choices for the similarity and coherence computations. The shorthand notation for the combinations in the table is as follows: *prior*: popularity prior; *r-prior*: popularity prior with robustness test; *sim-k*: keyphrase based similarity measure; *sim-s*: syntax-based similarity; *coh*: graph coherence; *r-coh*: graph coherence with robustness test.

The shorthand names for competitors are: *Cuc*: (Cucerzan07) similarity measure; *Kul_s*: (Kulkarni09) similarity measure only; *Kul_sp*: Kul_s combined with plus popularity prior; *Kul_CI*: Kul_sp combined with coherence. All coherence methods use the Milne-Witten inlink overlap measure mw_coh.

The most important measure is macro/micro precison@1.0, which corresponds to the overall correctness of the methods for all mentions that are assigned to an entity in the ground-truth data. Our *sim-k* precision is already very good. Combining it with the syntax-based similarity improves micro-averaged precision@1.0, but the macro-averaged results are a bit worse. Thus, the more advanced configurations of AIDA did not use syntax-based similarity. Unconditionally combining *prior* and *sim-k* degrades the quality, but including the prior robustness test (*r-prior sim-k*) improves the results significantly. The precision for our best method, the prior- and coherence-tested Keyphrase-based mention-entity similarity (*r-prior sim-k r-coh*), significantly outperforms all competitors (with a p-value of a paired t-test < 0.01). Our macro-averaged precision@1.0 is 81.91%, whereas *Kul_CI* only achieves 76.74%. Even *r-prior sim-k*, without any coherence, significantly outperforms

[1] 2 of the 231documents in the original test set could not be processed by *Kul_CI* due to memory limitations. All results are given for the subset, for the sake of comparability. Results for the complete set are available on our website.

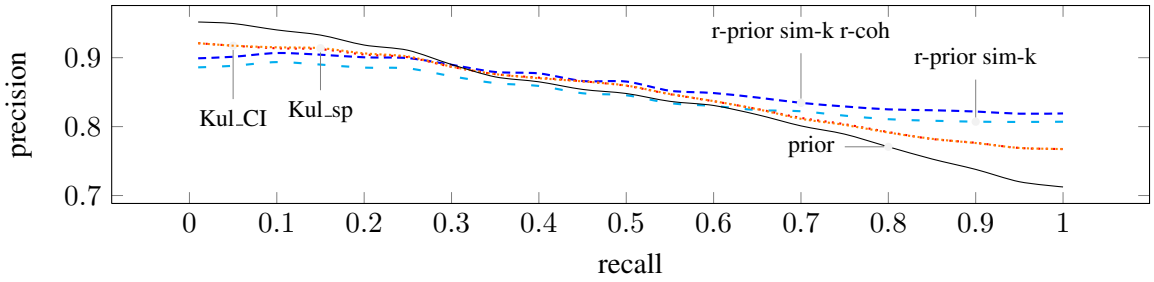


Figure 2: Experimental results on *CoNLL*: precision-recall curves

Kul_CI (with coherence) with a p-value of < 0.01. In micro-average precision@1.0, the differences are even higher, showing that we perform better throughout all documents.

The macro-averaged precision-recall curves in Figure 2 show that the best AIDA method performs particularly well in the tail of high recall values. The MAP underlines the robustness of our best methods.

The high MAP for the *prior* method is because we rank by mention-entity edge weight; for *prior* this is simply the prior probability. As the prior is most probably correct for mentions with a very high prior for their most popular entity (by definition), the initial ranking of the prior is very good, but drops more sharply. We believe that the main difficulty in named entity disambiguation lies exactly in the "long tail" of not-so-prominent entities.

We also tried the (Milne08) web service on a subset of our test collection, but this was obviously geared for Wikipedia linkage and performed poorly.

6.3 Discussion

Our keyphrase-based similarity measure performs better than the *Kul_s* measure, which is a combination of 4 different entity contexts (abstract tokens, full text tokens, inlink anchor tokens, inlink anchor tokens + surrounding tokens), 3 similarity measures (Jaccard, dot product, and tf.idf cosine similarity), and the popularity prior. Adding the prior to our similarity measure by linear combination degrades the performance. We found that our measure already captures a notion of popularity because popular entities have more keyphrases and can thus accumulate a higher total score. The popularity should only be used when one entitiy has a very high probability, and introducing the robustness test for the prior achieved this, improving on both our similarity and *Kul_sp*.

Unconditionally adding the notion of coherence among entities improves the micro-average precision, but not the macro-average. Investigating potential problems, we found that the coherence can be led astray when parts of the document form a coherent cluster of entities, and other entities are then forced to be coherent to this cluster. To overcome this issue, we introduced the coherence robustness test, and the results with *r-coh* show that it makes sense to fix an entity for a mention when the prior and similarity are in reasonable agreement. Adding this coherence test leads to a signigicant (p-value < 0.05) improvement over the non-coherence based measures in both micro- and macro-average precision. Our experiments showed that when adding this coherence test, around $\frac{2}{3}$ of the mentions are solved using local similarity only and are assigned an entity before running the graph algorithm. In summary, we observed that the AIDA configuration with *r-prior*, keyphrase-based *sim-k*, and *r-coh* significantly outperformed all competitors.

7 Conclusions and Future Work

The AIDA system provides an integrated NED method using popularity, similarity, and graph-based coherence, and includes robustness tests for self-adaptive behavior. AIDA performed significantly better than state-of-the-art baselines. The system is fully implemented and accessible online (`http://www.mpi-inf.mpg.de/yago-naga/aida/`). Our future work will consider additional semantic properties between entities (types, memberOf/partOf, etc.) for further enhancing the coherence algorithm.

Acknowledgements

This work has been partially supported by the German Science Foundation (DFG) through the Cluster of Excellence on "Multimodal Computing and Interaction" and the European Union through the 7th Framework IST Integrated Project "LivingKnowledge" (no. 231126). We also thank Mauro Sozio for the discussion on the graph algorithm.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, Zachary G. Ives: DBpedia: A Nucleus for a Web of Open Data. ISWC 2007

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, Oren Etzioni: Open Information Extraction from the Web. IJCAI 2007

Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: Compression techniques. WWW 2004, software at `http://webgraph.dsi.unimi.it/`

Razvan C. Bunescu, Marius Pasca: Using Encyclopedic Knowledge for Named entity Disambiguation. EACL 2006

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. AAAI 2010

Silviu Cucerzan: Large-Scale Named Entity Disambiguation Based on Wikipedia Data. EMNLP-CoNLL 2007

AnHai Doan, Luis Gravano, Raghu Ramakrishnan, Shivakumar Vaithyanathan. (Eds.). Special issue on information extraction. SIGMOD Record, 37(4), 2008.

Jenny Rose Finkel, Trond Grenager, Christopher Manning: Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. ACL 2005, software at `http://nlp.stanford.edu/software/CRF-NER.shtml`

Xianpei Han, Jun Zhao: Named entity disambiguation by leveraging wikipedia semantic knowledge. CIKM 2009.

Johannes Hoffart, Fabian Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, Gerhard Weikum: YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. Demo Paper, WWW 2011, data at `http://www.mpi-inf.mpg.de/yago-naga/yago/`

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, Soumen Chakrabarti: Collective annotation of Wikipedia entities in web text. KDD 2009

James Mayfield et al.: Corss-Document Coreference Resolution: A Key Technology for Learning by Reading. AAAI Spring Symposium on Learning by Reading and Learning to Read, 2009.

Diane McCarthy. Word Sense Disambiguation: An Overview. Language and Linguistics Compass 3(2): 537-558, Wiley, 2009

Rada Mihalcea, Andras Csomai: Wikify!: Linking Documents to Encyclopedic Knowledge. CIKM 2007

David N. Milne, Ian H. Witten: Learning to Link with Wikipedia. CIKM 2008

Ndapandula Nakashole, Martin Theobald, Gerhard Weikum: Scalable Knowledge Harvesting with High Precision and High Recall. WSDM 2011

Roberto Navigli: Word sense disambiguation: A survey. ACM Comput. Surv., 41(2), 2009

Hien T. Nguyen, Tru H. Cao: Named Entity Disambiguation on an Ontology Enriched by Wikipedia. RIVF 2008

Erik F. Tjong Kim Sang, Fien De Meulder: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. CoNLL 2003

Mauro Sozio, Aristides Gionis: The Community-search Problem and How to Plan a Successful Cocktail Party. KDD 2010

Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum: YAGO: a Core of Semantic Knowledge. WWW 2007

Fabian Suchanek, Mauro Sozio, Gerhard Weikum: SOFIE: a Self-Organizing Framework for Information Extraction. WWW 2009

Bilyana Taneva, Mouna Kacimi, and Gerhard Weikum: Finding Images of Rare and Ambiguous Entities. Technical Report MPI-I-2011-5-002, Max Planck Institute for Informatics, 2011.

Stefan Thater, Hagen Fürstenau, Manfred Pinkal. Contextualizing Semantic Representations using Syntactically Enriched Vector Models. ACL 2010

Michael L. Wick, Aron Culotta, Khashayar Rohanimanesh, Andrew McCallum: An Entity Based Model for Coreference Resolution. SDM 2009: 365-376

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, Ji-Rong Wen: StatSnowball: a Statistical Approach to Extracting Entity Relationships. WWW 2009

A Cascaded Classification Approach to Semantic Head Recognition

Lukas Michelbacher Alok Kothari Martin Forst[†]
Christina Lioma Hinrich Schütze
Institute for NLP
University of Stuttgart
{michells,kotharak,liomaca}@ims.uni-stuttgart.de
[†]Microsoft
martin.forst@microsoft.com

Abstract

Most NLP systems use tokenization as part of preprocessing. Generally, tokenizers are based on simple heuristics and do not recognize multi-word units (MWUs) like *hot dog* or *black hole* unless a precompiled list of MWUs is available. In this paper, we propose a new cascaded model for detecting MWUs of arbitrary length for tokenization, focusing on noun phrases in the physics domain. We adopt a classification approach because – unlike other work on MWUs – tokenization requires a completely automatic approach. We achieve an accuracy of 68% for recognizing non-compositional MWUs and show that our MWU recognizer improves retrieval performance when used as part of an information retrieval system.

1 Introduction

Most NLP systems use tokenization as part of preprocessing. Generally, tokenizers are based on simple heuristics and do not recognize multi-word units (MWUs) like *hot dog* or *black hole*. Our long-term goal is to build MWU-aware tokenizers that are used as part of the standard toolkit for NLP preprocessing alongside part-of-speech and named-entity tagging.

We define an MWU as a sequence of words that has properties that cannot be inferred from the component words (cf. e.g. Manning and Schütze (1999, Ch. 5), Sag et al. (2002)). The most important of these properties is non-compositionality, the fact that the meaning of a phrase cannot be predicted from the meanings of its component words. For example, a *hot dog* is not a hot animal but a sausage in a bun and a *black hole* in astrophysics is a region of space with special properties, not a dark cavity.

The correct recognition of MWUs is an important building block of many NLP tasks. For example, in information retrieval (IR) the query *hot dog* should not retrieve documents that only contain the words *hot* and *dog* individually, outside of the phrase *hot dog*.

In this study, we focus on noun phrases in the physics domain. For specialized domains such as physics, adaptable and reliable MWU recognition is of particular importance because comprehensive and up-to-date lists of MWUs are not available and would have to be created by hand. We chose noun phrases because domain-specific terminology is commonly encoded in noun phrase MWUs; other types of phrases – e.g., verb constructions – rarely give rise to fixed domain-specific multi-word sequences that should be treated as a unit.

We cast the task of MWU tokenization as *semantic head recognition* in this paper. The importance of syntactic heads for many NLP tasks is generally accepted. For example, in coreference resolution identity of syntactic heads is predictive of coreference; in parse disambiguation, the syntactic head of a noun phrase is a powerful feature for resolving attachment ambiguities. However, in all of these cases, the syntactic head is only an approximation of the information that is really needed; the underlying assumption made when using the syntactic head as a substitute for the entire phrase is that the syntactic head is representative of the phrase. This is not the case when the phrase is non-compositional.

We define the *semantic head* of a noun phrase as the non-compositional part of a phrase. Semantic heads would serve most NLP tasks better than syntactic heads. For example, a coreference resolution system is misled if it looks at syntactic heads to de-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 793–803,
Edinburgh, Scotland, UK, July 27–31, 2011.

termine possible coreference of *a hot dog ... the dog* in *I first ate a hot dog and then fed the dog*. This is not the case for a system that makes the decision based on the semantic heads *hot dog* of *a hot dog* and *dog* of *the dog*.

The specific NLP application we evaluate in this paper is information retrieval. We will show that semantic head recognition improves the performance of an information retrieval system.

We introduce a cascaded classification framework for recognizing semantic heads that allows us to treat noun phrases of arbitrary length. We use a number of previously proposed features for recognizing non-compositionality and semantic heads. In addition, we compare three features that measure contextual similarity.

Our main contributions in this paper are as follows. First, we introduce the notion of semantic head, in analogy to syntactic head, and propose semantic head recognition as a new component of NLP preprocessing. Second, we develop a cascaded classification framework for semantic head recognition. Third, we investigate the utility of contextual similarity for detecting non-compositionality and show that it significantly enhances a baseline semantic head recognizer. However, we also identify a number of challenges of using contextual similarity in high-confidence semantic head recognition. Fourth, we show that our approach to semantic head recognition improves the performance of an IR system.

Section 2 discusses previous work. In Section 3 we introduce semantic heads and present our cascaded model for semantic head recognition. In Section 4, we describe our data and three different measures of contextual similarity. Section 5 introduces the classifier and its features. Section 6 presents classification results and discussion. Section 7 describes the information retrieval experiments. In Section 8 we present our conclusions.

2 Related Work

While there is a large number of publications on MWUs and collocation extraction, the general problem of automatic MWU detection for the specific purpose of tokenization has not been investigated before to our knowledge.

The classic approach to identifying collocations and MWUs is to apply statistical association measures (AMs) to n-grams extracted from a corpus – often combined with various linguistic heuristics and other filters, resulting in candidate lists. Choueka (1988) and the XTRACT system (Smadja, 1993) are well-known examples of this approach.

More recent approaches such as Pecina (2010) and Ramisch et al. (2010) combine classifiers with association measures. Although our approach is classification-based as well, our data set has a more realistic size than Pecina (2010)'s (1 billion words vs 1.5 million words) and we work on noun phrases of arbitrary length (instead of just bigrams). The `mwetoolkit`[1] by Ramisch et al. (2010) aims to be a software package for lexicographers and its features are limited to a small set of association measures that do not consider marginal frequencies. Neither of these two studies includes evaluation in the context of an application.

Lin (1999) defines a decision criterion for non-compositional phrases based on the change in the mutual information of a phrase when substituting one word for a similar one based on an automatically constructed thesaurus. The method reaches 15.7% precision and 13.7% recall.

In terms of the extraction of domain-specific MWUs, cross-language methods have been proposed that make use of the fact that an MWU in one language might be expressed as a single word in another. Caseli et al. (2009) utilize word alignments in a parallel corpus; Attia et al. (2010) exploit the links between article names of different-language Wikipedias to search for many-to-one translations. We did not pursue a cross-language approach because we strive for a self-contained method of MWU recognition that operates on a single textual resource.

Non-compositionality and distributional semantics. In recent years, a number of studies have investigated the relationship between distributional semantics and non-compositionality. These studies compute the similarity between words and phrases represented as semantic vectors in a word space model. A semantic vector of a word is the accumulation of the particular contexts in which the word

[1] `http://sourceforge.net/projects/mwetoolkit/`

appears. The underlying idea is similar to Lin's: the meaning of a non-compositional phrase somehow deviates from what one would expect given the semantic vectors of parts of the phrase. The standard measure to compare semantic vectors is cosine similarity. The questions that arise are (i) which vectors to compare, (ii) how to combine the vectors of the parts and (iii) from what point on a certain dissimilarity indicates non-compositionality. To our knowledge, there are no generally accepted answers to these questions.

Regarding (i), Schone and Jurafsky (2001) compare the semantic vector of a phrase p and the vectors of its component words in two ways: one includes the contexts of p in the construction of the semantic vectors of the parts and one does not. Regarding (ii), they suggest weighted or unweighted sums of the semantic vectors of the parts.

Baldwin et al. (2003) investigate semantic decomposability of noun-noun compounds and verb constructions. They address (i) by comparing the semantic vectors of phrases with the vectors of their parts *individually* to detect meaning changes; e.g., they compare *vice president* to *vice* and *president*.

We propose a new method that compares phrases with their alternative phrases, in the spirit of Lin (1999)'s substitution approach (see Section 4.3). Our rationale is that context features should be based on contexts that are syntactically similar to the phrase in question.

With respect to (iii), the above-mentioned studies use ad hoc thresholds to separate compositional and non-compositional phrases but do not offer a principled decision criterion.[2] In contrast, we train a statistical classifier to learn a decision criterion.

There is a larger body of work concerning non-compositionality which revolves around the problem of literal (compositional) vs. non-literal (non-compositional) usage of idiomatic verb constructions like *to break the ice* or *to spill the beans*. Some studies approach the problem with semantic vector comparisons in the style of Schone and Jurafsky (2001), e.g Katz and Giesbrecht (2006) and Cook et al. (2007). Other approaches use word-alignment (e.g. Moirón and Tiedemann (2006)) or a combination of heuristic and linguistic features (e.g. Diab and Bhutada (2009), Li and Sporleder (2010)). Even though there is some methodological overlap between our approach and some of the verb-oriented studies, we believe that verb constructions have properties that are quite different from noun phrases. For example, our definition of alternative vector relies on the fact that most noun phrase MWUs are fixed and exhibit no syntactic variability. In contrast, verb constructions are often discontinuous.

The motivation for most work on MWU detection is lexicography, terminology extraction or the creation of machine-readable dictionaries. Our motivation – tokenization in a preprocessing setting – is different from this earlier work.

3 Semantic Heads and Cascaded Model

We cast the task of MWU tokenization as *semantic head recognition* in this paper. We define the semantic head of a noun phrase as the *largest non-compositional part of the phrase that contains the syntactic head*. For example, *black hole* is the semantic head of *unusual black hole* and *afterglow* is the semantic head of *bright optical afterglow*; in the latter case syntactic and semantic heads coincide.

Semantic heads would serve most NLP tasks better than syntactic heads. The attachment ambiguity of the last noun phrase in *he bought the hot dogs in a packet* can be easily resolved for the semantic head *hot dogs* (food is often in a packet), but not as easily for the syntactic head *dogs* (dogs are usually not in packets). Indeed, we will show in Section 7 that semantic head recognition improves the performance of an IR system.

The semantic head is either a single noun or a non-compositional noun phrase. In the latter case, the modifier(s) introduce(s) a non-compositional, unpredictable shift of meaning; *hot* shifts the meaning of *dog* from live animal to food. In contrast, the compositional meaning shift caused by *small* in *small dog* is transparent. The semantic head always contains the syntactic head; for compositional phrases, syntactic head and semantic head are identical.

To determine the semantic head of a phrase, we use a cascaded classification approach. The cascade

[2] Lin (1999) uses a well-defined criterion but his approach is not based on vector similarity.

(1)		*neutron*	**star**	
(2)	*unusual*	black	**hole**	
(3)	*bright*	optical	**afterglow**	
(4)		*small*	**moment**	of inertia

Figure 1: Example phrases with modifiers. Peripheral elements are set in italics, syntactic heads in bold.

comes into play in all aspects of our study: the rating experiments with human subjects, data extraction, feature design and classification itself.

We need a cascade because we want to recognize the semantic head in noun phrases of arbitrary length. The starting point is a phrase of length n: $p = w_1 \dots w_n$. We distinguish between the syntactic head of a phrase and the remaining words, the modifiers. Figure 1 shows phrases of varying syntactic complexity. The syntactic head is marked in bold. The model accommodates pre-nominal modifiers as in examples (1) through (3) and post-nominal modifiers like PPs in example (4).

Among the modifiers, there is a distinguished element, the *peripheral element* u (italicized in the examples). The remaining words are called the *rest* v. We can now represent any phrase p as $p = uv$.[3] The element u is always the outermost modifier. *of*-PPs are treated as a single modifier and they take precedence over pre-nominal modification because this analysis is dominant in our gold standard data. This means that in the phrase *small moment of inertia*, *small* (and not *of inertia*) is the peripheral element u.

Cascaded classification then operates as shown in Figure 2. In each iteration, the classifier decides whether the relation between the current peripheral element u and the rest v is compositional (C) or non-compositional (NC). If the relation is NC, processing stops and uv is returned as the semantic head of p. If the relation is compositional, u is discarded and classification continues with v as the new input phrase, which again is represented in the form $u'v'$. In case there is no more peripheral element u, i.e. the new v is a single word, it is returned as the semantic head of p.

Table 1 shows two examples. For the fully compositional phrase *bright optical afterglow*, the process runs all the way down to the syntactic head *afterglow* which is also the semantic head. In the second case, the process stops earlier, in step 2, because the classifier finds that the relation between *moment* and *of inertia* is NC. This means that the semantic head of *small moment of inertia* is *moment of inertia*.

[3] We use the abstract representation $p = uv$ even though u can appear after v in the surface form of p.

function $recognize\_semantic\_head(p)$
  $u \leftarrow peripheral(p)$
  $v \leftarrow rest(p)$
  while $decision(u, v) \neq NC$ **do**
    $u \leftarrow peripheral(v)$
    if $u = \emptyset$ **then**
      return v
    $v \leftarrow rest(v)$
  return uv

Figure 2: Cascaded classification of p

step	u	v	decision
1	*bright*	optical **afterglow**	C
2	*optical*	**afterglow**	C
3	$\emptyset$	**afterglow**	
1	*small*	**moment** of inertia	C
2	*of inertia*	**moment**	NC

Table 1: Cascaded decision processes

4 Corpus and Feature Definitions

4.1 Candidate phrases

As our corpus, we use the iSearch collection, a one billion word collection of documents from the physics domain (Lykke et al., 2010). We tokenized the collection by splitting on white space and adding sentence boundaries and part-of-speech tags to the output. With part-of-speech information, the identification of MWU candidates is easy, fast and reliable.

We extracted all noun phrases from the collection that consist of a head noun with up to four modifiers – almost all domain-specific terminology in our collection is captured by this pattern. The pre-nominal modifiers can be nouns, proper nouns, adjectives or cardinal numbers.

The baseline accuracy of a classifier that always chooses compositionality is very high ($> 90\%$) for

	$V = v$	$V \neq v$	
$U = u$	O_{11}	O_{12}	$= R_1$
$U \neq u$	O_{21}	O_{22}	$= R_2$
	$= C_1$	$= C_2$	$= N$

Table 2: 2-by-2 contingency tables with observed and marginal frequencies

phrases of the type *[noun] of the/a [noun] (sg.)* (e.g. *rest of the paper*) and *[noun] of [noun] (pl.)* (e.g. *series of papers*). We therefore restrict postnominal modifiers to prepositional phrases with the word *of* followed by a non-modified, indefinite, singular noun, e.g., *speed of light* or *moment of inertia*.

Out of all phrases extracted with part-of-speech patterns, we keep only the ones that appear more often than 50 times because it is hard to compute reliable features for less frequent phrases. All experiments were carried out with lemmatized word forms. We refer to lemmas as words if not noted otherwise.

4.2 Association measures

Statistical association measures are frequently used for MWU detection and collocation extraction (e.g. Schone and Jurafsky (2001), Evert and Krenn (2001), Pecina (2010)).

We use all measures used by Schone and Jurafsky (2001) that can be derived from a phrase's contingency table. These measures are Student's t-score, z-score, χ^2, pointwise mutual information (MI), Dice coefficient, frequency, log-likelihood (G^2) and symmetric conditional probability.

We define the AMs in Table 3 based on the notation for the contingency table shown in Table 2 (cf. Evert (2004)). O_{ij} is observed frequency and $E_{ij} = \frac{R_i C_j}{N}$ expected frequency.

The AMs are designed to deal with two random variables U and V that traditionally represent single words. In our model, we use U to represent peripheral elements u and V for rests v.

association measure	formula
student's t-score (am_t)	$\frac{O_{11}-E_{11}}{\sqrt{O_{11}}}$
z-score (am_z)	$\frac{O_{11}-E_{11}}{\sqrt{E_{11}}}$
chi-square (am_{χ^2})	$\sum_{i,j} \frac{(O_{ij}-E_{ij})^2}{E_{ij}}$
pointwise mutual information (am_{MI})	$\log \frac{O_{11}}{E_{11}}$
Dice coefficient (am_D)	$\frac{2O_{11}}{R_1+C_1}$
frequency (am_f)	O_{11}
log-likelihood (am_{G^2})	$2\sum_{i,j} O_{ij} \log \frac{O_{ij}}{E_{ij}}$
symmetric conditional probability (am_{scp})	$\frac{{O_{11}}^2}{R_1 C_1}$

Table 3: Association measures

4.3 Word space model

As our baseline, we use two methods of comparing semantic vectors: *sj1* and *sj2*, both introduced by Schone and Jurafsky (2001). They experimented with variants of sj1 and sj2, but found no large differences. In addition, we introduce our own approach *alt*.

Method sj1 compares the semantic vector of a phrase p with the sum of the vectors of its parts. Method sj2 is like sj1, except the contexts of p are not part of the semantic vectors of the parts. Method alt compares the semantic vector of a phrase with its alternative vector. In the definitions below, s represents a vector similarity measure, $w(p)$ a general semantic vector of a phrase p and $w^*(w_i)$ the semantic vector of a part w_i of a phrase p that does not include the contexts of occurrences of w_i that were part of p itself.

sj1 $s(w(\textit{black hole}),\ w(\textit{black}) + w(\textit{hole}))$

sj2 $s(w(\textit{black hole}),\ w^*(\textit{black}) + w^*(\textit{hole}))$

alt $s(w(\textit{black hole}), \sum_u w(u,\ \textit{hole}));\ u \neq \textit{black}$

For the third comparison, we build the *alternative vector* as follows. For a phrase $p = uv$ with peripheral element u and rest v, we call the phrase

$p' = u'v$ an *alternative phrase* if the rest v is the same and $u' \neq u$. E.g., *giant star* is an alternative phrase of *neutron star* and *isolated neutron star* is an alternative of *young neutron star*. The alternative vector of p is then the semantic vector that is computed from the contexts of all of p's alternative phrases. The alternative vector is a representation of the contexts of v except for those modified by u. This technique bears resemblance to the substitution approach of Lin (1999). The difference is that he relies on a similarity thesaurus for substitution and monitors the change in mutual information for each substitution individually whereas we substitute with general alternative modifiers and combine the alternative contexts into one vector for comparison.

Previous work has compared the semantic vector of a phrase with the vectors of its components. Our approach is more "head-centric" and only compares phrases in the same syntactic configuration. Our question is: Is the typical context of the head *hole* if it occurs with a modifier that is not *black* different from when it occurs with the modifier *black*?

We used a bag-of-words model and a window of ± 10 words for contexts to create semantic vectors. We only kept the content words in the window which we defined as words that are tagged as either a noun, verb, adjective or adverb. To add information about the variability of syntactic contexts in which phrases occur, we add the words immediately before and after the phrase with positional markers (-1 and $+1$, respectively) to the vector. These words were not subject to the content-word filter. The dimensionality of the vectors is then $3V$ where V is the size of the vocabulary: V dimensions each for bag-of-words, left and right syntactic contexts. We did not include vectors for the stop word *of* for sj1 and sj2.

4.4 Non-compositionality judgments

Since the domain of the corpus is physics, highly specialized vocabulary had to be judged. We employed domain experts as raters (one engineering and two physics graduate students).

In line with the cascaded model, the raters where asked to identify the semantic head of each candidate phrase. If at least two raters agreed on a semantic head of a phrase we made this choice the semantic head in the gold standard. The final gold standard comprises 1560 phrases.

We computed raw agreement of each rater with the gold standard as the percentage of correctly recognized semantic heads – this is the task that the classifier addresses. Agreement is quite high at 86.5%, 88.3% and 88.5% for the three raters. In addition, we calculated chance-corrected agreement with Cohen's κ on the first decision task against the gold standard (see Section 6). As expected, agreement decreases, but is still substantial at 74.0%, 78.2% and 71.8% for the three raters.

5 Classifier

We use the Stanford maximum entropy classifier for our experiment.[4] We randomly split the data into a training set of 1300 and a held-out test set of 260 pairs.

We use the eight AMs and the cosine similarities sim_{sj1}, sim_{sj2} and sim_{alt} described in Section 4.3 as features for the classifier. Cosine similarity should be small if a phrase is non-compositional and large if it is compositional. In other words, if the contexts of the candidate phrase are too dissimilar to the contexts of the sum of its parts or to the alternative phrases, then we suspect non-compositionality.

Feature values are binned into 5 bins. We applied a log transformation to the four AMs with large values: am_f, am_{G^2}, am_{χ^2} and am_z. For our application there is little difference between statistical significance at $p < .001$ and $p < .00001$. The log transformation reduces the large gap in magnitude between high significance and very high significance. If co-occurrence of u and v in uv is below chance, then we set the association scores to 0 since this is an indication of compositionality (even if it is highly significant).

Since AMs have been shown to be correlated (e.g. Pecina (2010)), we first perform feature selection on the AM features. We tested accuracy of all $2^r - 1$ non-empty combinations of the $r = 8$ AM features on the task of deciding whether the first decision during the classification of a phrase was C or NC. We then selected those AM features that were part of at least one top 10 result in each fold. Those features were am_t, am_f and am_{scp}.

The main experiment combines these three se-

[4] http://nlp.stanford.edu/software/classifier.shtml

lected AM features with all possible subsets of context features. We train on the 1300-element training set and test on the 260-element test set.

6 Results and Discussion

We ran three evaluation modes: *dec-1st*, *dec-all*, and *semh*. Mode *dec-1st* only evaluates the first decision for each phrase; the baseline in this case is .554 since 55.4% of the first decisions are C. In mode *dec-all*, we evaluate all decisions that were made in the course of recognizing the semantic head. This mode emphasizes the correct recognition of semantic heads in phrases where multiple correct decisions in a row are necessary. We define the confidence for multi-decision classification as the product of the confidence values of all intermediate decisions. There is no obvious baseline for *dec-all* because the number of decisions depends on the classifier – a classifier whose first decision on a four-word phrase is NC makes one decision, another one may make three. The mode *semh* evaluates how many semantic heads were recognized correctly. This mode directly evaluates the task of semantic head recognition. The baseline for *semh* is the tokenizer that always returns the syntactic head; this baseline is .488.[5] Table 4 shows 8×3 runs, corresponding to the three modes tested on the AM features (am_t, am_f, and am_{scp}) and the eight possible subsets of the three context features.

For all modes, the best result is achieved with base AMs combined with the sim_{alt} feature; the accuracies are .692, .703 and .680. The improvements over the baselines (for *dec-1st* and *semh*) are statistically significant at $p < .01$ (binomial test, $n = 260$).

For *semh*, accuracy without any context features is .603; this is significantly better than the .488 baseline ($p < .01$). Performance with only the base AM features is significantly lower than the best context feature experiment (.680) at $p < .01$ and significantly lower than the worst context feature experiment (.653) at $p < .1$. However, the differences between the context feature runs are not significant.

When the semantic head recognizer processes a phrase, there are four possible results. Result r_{semh}: the semantic head is correctly recognized and it is distinct from the syntactic head. Result r_{synth}: the semantic head is correctly recognized and it is identical to the syntactic head. Result r_+: the semantic head is not correctly recognized because the cascade was stopped too early, i.e., a compositional modifier that should have been removed was kept. Result r_-: the semantic head is not correctly recognized because the cascade was stopped too late, i.e., a modifier causing a non-compositional meaning shift was removed. Table 5 shows the distribution of result types. It shows that r_+ is the more common error: the classifier more often regards compositional relations as non-compositional than vice versa.

type	freq	definition
r_{semh}	92	sem. head correct ($\neq$ synt. head)
r_{synth}	85	sem. head correct (= synt. head)
r_+	48	sem. head too long
r_-	35	sem. head too short
all	260	

Table 5: Distribution of result types

Table 6 shows the top 20 classifications where the semantic head was not the same as the syntactic head sorted by confidence in descending order. In the third column "phrase ..." we list the candidates with semantic heads in bold. The columns to the right show the predicted semantic head and the feature values. All five errors in the list are of type r^+.

Two r^+ phrases are *schematic view* and *many others*. The two phrases are clearly compositional and the classifier failed even though the context feature points in the direction of compositionality with a value greater than .5. It can be argued that *many others* is a trivial example that does not require complex machinery to be identified as compositional, e.g. by using a stop list. We included it in the analysis since we want to be able to process arbitrary phrases without additional hand-crafted resources.

Another incorrect classification occurs with the phrase *massive star birth*[6] for which *star birth* was annotated as the semantic head. Here we have a case where the peripheral element *massive* does not mod-

[5] The baseline could be improved with simple heuristics, e.g. "uv contains capital letter" $\rightarrow NC$. However, this feature only results in a 2% improvement compared to the baseline.

[6] i.e. the birth of a massive star, a certain type of star with very high mass

mode	baseline	context feature	context feature subsets							
		sim_{alt}	-	•	•	•	•	-	-	-
		sim_{sj1}	-	-	•	–	•	•	-	•
		sim_{sj2}	-	-	-	•	•	-	•	•
dec-1st	.554		.604	.692	.669	.685	.677	.654	.654	.662
dec-all	-		.615	.703	.681	.696	.688	.666	.669	.675
semh	.488		.603	**.680**	.657	.673	.665	.653	.653	.661

Table 4: Performance for base AM features plus context feature subsets. A '•' indicates the use of the corresponding context feature.

ify the syntactic head *birth* but *massive star* is itself a complex modifier. In the test set, 5% of the phrases exhibit structural ambiguities of this type. Our system cannot currently deal with this phenomenon.

The remaining r^+ phrases are *peculiar velocity* and *local group*. However, Wikipedia lists both phrases with an individual entry defining the former as *the true velocity of an object, relative to a rest frame*[7] and the latter as *the group of galaxies that includes Earth's galaxy, the Milky Way*[8]. Both definitions provide evidence for non-compositionality since the velocity is not peculiar (as in strange) and the scope of *local* is not clear without further knowledge. Arguably, in these cases our method chose a justifiable semantic head, but the raters disagreed.[9]

For NLP preprocessing, it is acceptable to sacrifice recall and only make high-confidence decisions on semantic heads. A tokenizer that reliably detects a subset of MWUs is better than one that recognizes none. However, our attempts to use the sim_{alt} recognizer (bold in Table 4) in this way were not successful. Precision is .680 for confidence $>$ .7 and does not exceed .770 for higher confidence values.

To understand this effect, we analyzed the distribution of sim_{alt} scores. Surprisingly, moderate similarity between .4 and .6 is a more reliable indicator for NC than low similarity $<$ .3. Our intuition for using distributional semantics in Section 2 was that low similarity indicates non-compositionality. This does not seem to hold for the lowest similarity values possibly because they are often extreme cases in terms of distribution and frequency and then give rise to unreliable decisions. This means that the context features enhance the overall performance of the classifier, but they are unreliable and do not support the high-confidence decisions we need in NLP preprocessing.

For comparison, the classifier that only uses AM features achieves 90% precision at 14% recall with confidence $>$ .7 – although it has lower overall accuracy than the sim_{alt} recognizer. We are still in the process of analyzing these results and decided to use the AM-only recognizer for the IR experiment because it has more predictable performance.

In summary, the results show that, for the recognition of semantic heads, basic AMs offer a significant improvement over the baseline. We have shown that some wrong decisions are defensible even though the gold standard data suggests otherwise. Context features further increase performance significantly, but surprisingly, they are not of clear benefit for a high-confidence classifier that is targeted towards recognizing a smaller subset of semantic heads with high confidence.

7 Information Retrieval Experiment

Typically, IR systems do not process non-compositional phrases as one semantic entity, missing out on potentially important information captured by non-compositionality. This section illustrates one way of adjusting the retrieval process so that non-compositional phrases are processed as semantic entities that may enhance retrieval performance. The underlying hypothesis is that, given

[7] http://en.wikipedia.org/wiki/Peculiar_velocity

[8] http://en.wikipedia.org/wiki/Local_group

[9] Further evidence that *local group* is non-compositional is the fact that one of the domain experts annotated the phrase as non-compositional but was overruled by the other two.

c.	type	phrase (semantic head in bold)	predicted semantic head	am_t	am_f	am_{cp}	sim_{alt}
.99	r_{semh}	**ellipsoidal figure of equilibrium**	ellipsoidal figure of equilibrium	18.03	325	6.23e-01	.219
.99	r_{semh}	**point spread function**	point spread function	95.03	9056	2.33e-01	.529
.99	r_{+}	massive **star birth**	massive star birth	19.99	402	4.81e-03	.134
.98	r_{semh}	**high angular resolution imaging**	high angular resolution imaging	13.07	179	1.27e-03	.173
.98	r_{semh}	**integral field spectrograph**	integral field spectrograph	24.20	586	4.12e-02	.279
.98	r_{+}	local **group**	local group	153.54	24759	8.73e-03	.650
.98	r_{semh}	**neutral kaon system**	neutral kaon system	1.38	108	4.17e-03	.171
.97	r_{semh}	**IRAF task**	IRAF task	49.07	2411	2.96e-02	.517
.92	r_{semh}	**easy axis**	easy axis	44.66	2019	2.79e-03	.599
.89	r_{+}	schematic **view**	schematic view	40.56	1651	8.06e-03	.612
.87	r_{semh}	**differential resistance**	differential resistance	31.71	1034	6.38e-04	.548
.86	r_{semh}	**TiO band**	TiO band	36.84	1372	2.21e-03	.581
.86	r_{+}	many **others**	many others	97.76	9806	6.54e-03	.708
.86	r_{semh}	**VLBA observation**	VLBA observation	43.95	2004	9.35e-04	.648
.85	r_{+}	peculiar **velocity**	peculiar velocity	167.63	28689	2.37e-02	.800
.84	r_{semh}	**computation time**	computation time	43.80	1967	1.35e-03	.657
.83	r_{semh}	**Land factor**	Land factor	21.15	453	6.30e-04	.360
.83	r_{semh}	**interference filter**	interference filter	31.44	1002	1.27e-03	.574
.83	r_{semh}	**line formation calculations**	line formation calculations	14.20	203	1.96e-03	.381
.82	r_{semh}	**Wess-Zumino-Witten term**	Wess-Zumino-Witten term	9.60	94	8.12e-05	.291

Table 6: The 20 most confident classifications where the prediction is semantic head $\neq$ syntactic head. "c." = confidence

a query that contains a non-compositional phrase, boosting the retrieval weight of documents that contain this phrase will improve overall retrieval performance.

We do this boosting using Indri's[10] combination of the language modeling and inference network approaches (Metzler and Croft, 2004), which allows assigning different degrees of belief to different parts of the query. This belief can be drawn from any suitable external evidence of relevance. In our case, this source of evidence is the knowledge that certain query terms constitute a non-compositional phrase. Under this approach, and using the *#weight* and *#combine* operators for combining beliefs, the relevance of a document D to a query Q is computed as the probability that D generates Q, $P(Q|D)$:

$$P(Q|D) = \prod_{t \in Q} P(t|D)^{\frac{w_t}{W}} \qquad (W = \sum_{t \in Q} w_t) \quad (1)$$

where t is a term and w_t is the belief weight assigned to t. The higher w_t is, the higher the rank of documents containing t. In this work, we distinguish between two types of query terms: terms occurring in non-compositional phrases (Q_{nc}), and the remaining query terms (Q_c). Terms $t \in Q_{nc}$ receive belief weight w_{nc} and terms $t \in Q_c$ belief weight w_c, ($w_{nc} + w_c = 1$ and $w_{nc}, w_c \in [0, 1]$). To boost the ranking of documents containing non-compositional phrases, we increase w_{nc} at the expense of w_c. We estimate $P(t|D)$ in Eq. 1 using Dirichlet smoothing (Zhai and Lafferty, 2002).

We use Indri for indexing and retrieval without removing stopwords or stemming. This choice is motivated by two reasons: (i) We do not have a domain-specific stopword list or stemmer. (ii) Baseline performance is higher when keeping stopwords and without stemming, rather than without stopwords and with stemming.

We use the iSearch collection discussed in Section 4. It comprises 453,254 documents and a set of 65 queries with relevance assessments. To match documents to queries without any treatment of non-compositionality (baseline run), we use the Kullback-Leibler language model with Dirichlet smoothing (KL-Dir) (Zhai and Lafferty, 2002). We applied the preprocessing described

[10] `http://www.lemurproject.org/`

run	MAP	REC	P20
baseline	0.0663	770	0.1385
real NC	**0.0718**	**844**	**0.1538**
pseudo NC_1	0.0664	788	0.1385
pseudo NC_2	0.0658	782	0.1462
pseudo NC_3	0.0671	777	0.1477
pseudo NC_4	0.0681	807	0.1462
pseudo NC_5	0.0670	783	0.1423

Table 7: IR performance without considering non-compositionality (*baseline*), versus boosting real and pseudo non-compositionality (*real NC, pseudo* NC_i).

in Section 4 to the queries and identified non-compositional phrases with the base AM classifier from Section 5. Our approach for boosting the weight of these non-compositional phrases uses the same retrieval model enhanced with belief weights as described in Eq. 1 (real NC run). In addition, we include five runs that boost the weight of pseudo non-compositional phrases that were created randomly from the query text (pseudo NC runs). These pseudo non-compositional phrases have exactly the same length as the observed non-compositional phrases for each query. We measure retrieval performance in terms of mean average precision (MAP), precision at 20 (P20), and recall (REC, number of relevant documents retrieved – total is 2878). For each evaluation measure separately, we tune the following parameters and report the best performance: (i) the smoothing parameter μ of the KL-Dir retrieval model ($\mu \in \{100, 500, 800, 1000, 2000, 3000, 4000, 5000, 8000, 10000\}$, following Zhai and Lafferty (2002)); (ii) the belief weights $w_{nc}, w_c \in \{0.1, \ldots, 0.9\}$ in steps of 0.1 while preserving $w_{nc} + w_c = 1$ at all times.

Table 7 displays retrieval performance of our approach against the baseline and five runs with pseudo non-compositional phrases. We see a 9.61% improvement in the number of relevant retrieved documents over the baseline. MAP and P20 also show improvements. Our approach is better than any of the 5 random runs on all three metrics – the probability of getting such a good result by chance is $\frac{1}{2^5} < .05$, and thus the improvements are statistically significant. On doing a query-wise analysis of MAP scores, we find that large improvements over the baseline occur when a non-compositional phrase aligns with what the user is looking for. The system seems to retrieve more relevant documents in that case. E.g., the improvement in MAP is 0.0977 for query #19. The user was looking for *"articles ... on making tunable vertical cavity surface emitting laser diodes"* and *laser diodes* was one of the non-compositional phrases recognized. On the other hand, a decrease in MAP occurs for non-compositional phrases unrelated to the information need. In query #4 the user is looking for *"protein-protein interaction, the surface charge distribution of these proteins and how this has been investigated with Electrostatic Force Microscopy"* and though non-compositional phrases such as *Force Microscopy* are recognized, these do not reflect the core information need *"The proteins of interest are the Avidin-Biotin and IgG-anti-IgG systems"*.

8 Conclusion

We have presented an approach to improving tokenization in NLP preprocessing that is based on the notion of semantic head. Semantic heads are – in analogy to syntactic heads – the core meaning units of phrases that cannot be further semantically decomposed. To perform semantic head recognition for tokenization, we defined a novel cascaded model and implemented it as a statistical classifier that used previously proposed and new context features. We have shown that the classifier significantly outperforms the baseline and that context features increase performance. We reached an accuracy of 68% and argued that even a semantic head recognizer restricted to high-confidence decisions is useful – because reliably recognizing a subset of semantic heads is better than recognizing none. We showed that context features increase the accuracy of the classifier, but undermine the confidence assessments of the classifier, a result we are still analyzing. Finally, we showed that even in its preliminary current form the semantic head recognizer is able to improve the performance of an IR system.

Acknowledgments

This work was funded by DFG projects SFB 732 and WordGraph. We also thank the anonymous reviewers for their comments.

References

Mohammed Attia, Antonio Toral, Lamia Tounsi, Pavel Pecina, and Josef van Genabith. 2010. Automatic extraction of arabic multiword expressions. In *Proceedings of the 2010 Workshop on Multiword Expressions*, pages 19–27, Beijing, China. Coling 2010 Organizing Committee.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions*, pages 89–96, Sapporo, Japan. Association for Computational Linguistics.

Helena Caseli, Aline Villavicencio, André Machado, and Maria José Finatto. 2009. Statistically-driven alignment-based multiword expression identification for technical domains. In *Proceedings of the 2009 Workshop on Multiword Expressions*, pages 1–8, Singapore. Association for Computational Linguistics.

Yaacov Choueka. 1988. Looking for needles in a haystack. In *Proceedings of RIAO88*, pages 609–623.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the 2007 on Multiword Expressions*, pages 41–48, Prague, Czech Republic. Association for Computational Linguistics.

Mona Diab and Pravin Bhutada. 2009. Verb noun construction mwe token classification. In *Proceedings of the 2009 Workshop on Multiword Expressions*, pages 17–22, Singapore. Association for Computational Linguistics.

Stefan Evert and Brigitte Krenn. 2001. Methods for the qualitative evaluation of lexical association measures. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 188–195. Association for Computational Linguistics.

Stefan Evert. 2004. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Ph.D. thesis, Institut für maschinelle Sprachverarbeitung (IMS), Universität Stuttgart.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the 2006 Workshop on Multiword Expressions*, pages 12–19, Sydney, Australia. Association for Computational Linguistics.

Linlin Li and Caroline Sporleder. 2010. Linguistic cues for distinguishing literal and non-literal usages. In *Coling 2010: Posters*, pages 683–691, Beijing, China. Coling 2010 Organizing Committee.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 317–324, College Park, Maryland, USA. Association for Computational Linguistics.

Marianne Lykke, Birger Larsen, Haakon Lund, and Peter Ingwersen. 2010. Developing a test collection for the evaluation of integrated search. In *Advances in Information Retrieval, 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010. Proceedings*, pages 627–630.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Donald Metzler and W. Bruce Croft. 2004. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.*, 40(5):735–750.

B.V. Moirón and Jörg Tiedemann. 2006. Identifying Idiomatic Expressions Using Automatic Word-Alignment. In *Multi-Word-Expressions in a Multilingual Context*, page 33.

Pavel Pecina. 2010. Lexical association measures and collocation extraction. *Language Resources and Evaluation*, 44(1-2):138–158.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010. mwetoolkit: a framework for multiword expression identification. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15, Mexico City.

Patrick Schone and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 100–108, Pittsburgh, Pennsylvania, USA. Association for Computational Linguistics.

Frank Smadja. 1993. Retrieving collocations from text: Xtract. *Computational linguistics*, 19(1):143–177.

ChengXiang Zhai and John D. Lafferty. 2002. Two-stage language models for information retrieval. In *SIGIR*, pages 49–56. ACM.

Linking Entities to a Knowledge Base with Query Expansion

Swapna Gottipati
School of Information Systems
Singapore Management University
Singapore
swapnag.2010@smu.edu.sg

Jing Jiang
School of Information Systems
Singapore Management University
Singapore
jingjiang@smu.edu.sg

Abstract

In this paper we present a novel approach to entity linking based on a statistical language model-based information retrieval with query expansion. We use both local contexts and global world knowledge to expand query language models. We place a strong emphasis on named entities in the local contexts and explore a positional language model to weigh them differently based on their distances to the query. Our experiments on the TAC-KBP 2010 data show that incorporating such contextual information indeed aids in disambiguating the named entities and consistently improves the entity linking performance. Compared with the official results from KBP 2010 participants, our system shows competitive performance.

1 Introduction

When people read news articles, Web pages and other documents online, they may encounter named entities which they are not familiar with and therefore would like to look them up in an encyclopedia. It would be very useful if these entities could be automatically linked to their corresponding encyclopedic entries. This task of linking mentions of entities within specific contexts to their corresponding entries in an existing knowledge base is called *entity linking* and has been proposed and studied in the Knowledge Base Population (KBP) track of the Text Analysis Conference (TAC) (McNamee and Dang, 2009). Besides improving an online surfer's browsing experience, entity linking also has potential usage in many other applications such as normalizing entity mentions for information extraction.

The major challenge of entity linking is to resolve name ambiguities. There are generally two types of ambiguities: (1) Polysemy: This type of ambiguities refers to the case when more than one entity shares the same name. E.g. *George Bush* may refer to the 41st President of the U.S., the 43rd President of the U.S., or any other individual who has the same name. Clearly polysemous names cause difficulties for entity linking. (2) Synonymy: This type of ambiguities refers to the case when more than one name variation refers to the same entity. E.g. *Metro-Goldwyn-Mayer Inc.* is often abbreviated as *MGM*. Synonymy affects entity linking when the entity mention in the document uses a name variation not covered in the entity's knowledge base entry.

Intuitively, to disambiguate a polysemous entity name, we should make use of the context in which the name occurs, and to address synonymy, external world knowledge is usually needed to expand acronyms or find other name variations. Indeed both strategies have been explored in existing literature (Zhang et al., 2010; Dredze et al., 2010; Zheng et al., 2010). However, most existing work uses supervised learning approaches that require careful feature engineering and a large amount of training data. In this paper, we take a simpler unsupervised approach using statistical language model-based information retrieval. We use the KL-divergence retrieval model (Zhai and Lafferty, 2001) and expand the query language models by considering both the local contexts within the query documents and global world knowledge obtained from the Web.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 804–813,
Edinburgh, Scotland, UK, July 27–31, 2011.

Symbol	Description
Q	Query
D_Q	Query document
N_Q	Query name string
E	KB entity node
N_E	KB entity name string
D_E	KB entity disambiguation text
$\mathcal{S}_Q$	Set of alternate query name strings
$N_Q^{l,i}$	Local alternative name strings
N_Q^g	Global alternative name strings
$\mathcal{E}_Q$	Candidate KB entries for Q
θ_Q	Query Language Model
θ_Q^L	KB entry language model using local context from D_Q
θ_Q^G	KB entry language model using global knowledge
θ_Q^{L+G}	KB entry language model using local context and global knowledge
θ_{N_E}	KB entry language model with named entities only
$\theta_{N_E+D_E}$	KB entry language model with named entities and disambiguation text

Table 1: Notation

We evaluate our retrieval method with query expansion on the 2010 TAC-KBP data set. We find that our expanded query language models can indeed improve the performance significantly, demonstrating the effectiveness of our principled and yet simple techniques. Comparison with the official results from KBP participants also shows that our system is competitive. In particular, when no disambiguation text from the knowledge base is used, our system can achieve an overall 85.2% accuracy and 9.3% relative improvement over the best performance reported in KBP 2010.

2 Task Definition and System Overview

Following TAC-KBP (Ji et al., 2010), we define the entity linking task as follows. First, we assume the existence of a Knowledge Base (KB) of entities. Each KB entry E represents a unique entity and has three fields: (1) a name string N_E, which can be regarded as the official name of the entity, (2) an entity type T_E, which is one of {PER, ORG, GPE, UNKNOWN}, and (3) some disambiguation text D_E. Given a query Q which consists of a query name string N_Q and a query document D_Q where the name occurs, the task is to return a single KB entry to which the query name string refers or *Nil* if there is no such KB entry.

It is fairly natural to address entity linking by ranking the KB entries given a query. In this section we present an overview of our system, which consists of two major stages: a candidate selection stage to identify a set of candidate KB entries through name matching, and a ranking stage to link the query entity to the most likely KB entry. In both stages, we consider the query's local context in the query document and world knowledge obtained from the Web. It is important to note that the selection stage is based on string matching where the order of the word matters. It is different from the ranking stage where a probabilistic retrieval model based on bag-of-word representation is used. Our preliminary experiments demonstrate that without the first candidate selection stage the linking process results in low performance.

2.1 Selecting Candidate KB Entries

The first stage of our system aims to filter out irrelevant KB entries and select only a set of candidates that are potentially the correct match to the query. Intuitively, we determine whether two entities are the same by comparing their name strings. We therefore need to compare the query name string N_Q with the name string N_E of each KB entry. However, because of the name ambiguity problem, we cannot expect the correct KB entry to always have exactly the same name string as the query. To address this problem, we use a set of alternative name strings expanded from N_Q and select KB entries whose name

strings match at least one of them. These alternative name strings come from two sources: the query document D_Q and the Web.

First, we observe that some useful alternative name strings come from the query document. For example, a PER query name string may contain only a person's last name but the query document contains the person's full name, which is clearly a less ambiguous name string to use. Similarly, a GPE query name string may contain only the name of a city or town but the query document contains the state or province, which also helps disambiguate the query entity. Based on this observation, we do the following. Given query Q, let $\mathcal{S}_Q$ denote the set of alternative query name strings. Initially $\mathcal{S}_Q$ contains only N_Q. We then use an off-the-shelf NER tagger to identify named entities from the query document D_Q. For PER and ORG queries, we select named entities in D_Q that contain N_Q as a substring. For GPE queries, we select named entities that are of the type GPE, and we then combine each of them with N_Q. We denote these alternative name strings as $\{N_Q^{l,i}\}_{i=1}^{K_Q}$, where l indicates that these name strings come locally from D_Q and K_Q is the total number of such name strings. $\{N_Q^{l,i}\}$ are added to $\mathcal{S}_Q$. Figure 1 and Figure 2 show two example queries together with their $\mathcal{S}_Q$.

Sometimes alternative name strings have to come from external knowledge. For example, one of the queries we have contains the name string "AMPAS," and the query document also uses only this acronym to refer to this entity. But the full name of the entity, "Academy of Motion Pictures Arts and Sciences," is needed in order to locate the correct KB entry. To tackle this problem, we leverage Wikipedia to find the most likely official name. Given query name string N_Q, we check whether the following link exists: `http://en.wikipedia.org/`N_Q. If N_Q is an abbreviation, Wikipedia will redirect the link to the Wikipedia page of the corresponding entity with its official name. So if the link exists, we use the title of the Wikipedia page as another alternative name string for N_Q. We refer to this name string as N_Q^g to indicate that it is a global name variant. N_Q^g is also added to $\mathcal{S}_Q$. Figure 2 shows such an example.

For each name string N in $\mathcal{S}_Q$, we find KB entries whose name strings match N. We take the union of these sets of KB entries and refer to it as $\mathcal{E}_Q$. These are the candidate KB entries for query Q.

Query name string (N_Q): *Mobile*
Query document (D_Q): *The site is near Mount Vernon in the Calvert community on the Tombigbee River, some 25 miles (40 kilometers) north of Mobile. It's on a river route to the Gulf of Mexico and near Mobile's rails and interstates. Along with tax breaks and $400 million (euro297 million) in financial incentives, Alabama offered a site with a route to a Brazil plant that will provide slabs for processing in Mobile.*
Alternative Query Strings ($\mathcal{S}_Q$):
from local context: *Mobile, Mobile Mount Vernon, Mobile Calvert, Mobile River, Mobile Mexico, Mobile Alabama, Mobile Brazil*

Figure 1: An example *GPE* query from TAC 2010.

Query name string (N_Q): *Coppola*
Query document (D_Q): *I had no idea of all these semi-obscure connections, felicia! Alex Greenwald and Claire Oswalt aren't names I'm at all familiar with, but Jason Schwartzman I've heard of. Isn't he Sophia Coppola's cousin? I think I once saw a picture of him sometime ago*
Alternative Query Strings ($\mathcal{S}_Q$):
from local context: *Coppola, Sophia Coppola, Sofia Coppola*
from world knowledge(Wikipedia): *Sofia Coppola*

Figure 2: An example *PER* query from TAC 2010.

2.2 Ranking KB Entries

Given the candidate KB entries $\mathcal{E}_Q$, we need to decide which one of them is the correct match. We adopt the widely-used KL-divergence retrieval model, a statistical language model-based retrieval method proposed by Lafferty and Zhai (2001). Given a KB entry E and query Q, we score E based on the KL-divergence defined below:

$$s(E,Q) = -Div(\theta_Q \| \theta_E) = -\sum_{w \in V} p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_E)}. \quad (1)$$

Here θ_Q and θ_E are the query language model and the KB entry language model, respectively. A language model here is a multinomial distribution over words (i.e. a unigram language model). V is the vocabulary and w is a single word.

To estimate θ_E, we follow the standard maximum likelihood estimation with Dirichlet smooth-

ing (Zhai and Lafferty, 2004):

$$p(w|\theta_E) = \frac{c(w,E) + \mu p(w|\theta_C)}{|E| + \mu}, \quad (2)$$

where $c(w,E)$ is the count of w in E, $|E|$ is the number of words in E, θ_C is a background language model estimated from the whole KB, and μ is the Dirichlet prior. Recall that E contains N_E, T_E and D_E. We consider using either N_E only or both N_E and D_E to obtain $c(w,E)$ and $|E|$. We refer to the former estimated θ_E as θ_{N_E} and the latter as $\theta_{N_E+D_E}$.

To estimate θ_Q, typically we can use the empirical query word distribution:

$$p(w|\theta_Q) = \frac{c(w,N_Q)}{|N_Q|}, \quad (3)$$

where $c(w,N_Q)$ is the count of w in N_Q and $|N_Q|$ is the length of N_Q. We call this model the *original* query language model.

After ranking the candidate KB entries in $\mathcal{E}_Q$ using Equation (1), we perform entity linking as follows. First, using an NER tagger, we determine the entity type of the query name string N_Q. Let T_Q denote this entity type. We then pick the top-ranked KB entry whose score is higher than a threshold τ and whose T_E is the same as T_Q. The system links the query entity to this KB entry. If no such entry exists, the system returns *Nil*.

3 Query Expansion

We have shown in Section 2.1 that using the original query name string N_Q itself may not be enough to obtain the correct KB entry, and additional words from both the query document and external knowledge can be useful. However, in the KB entry selection stage, these additional words are only used to enlarge the set of candidate KB entries; they have not been used to rank KB entries. In this section, we discuss how to expand the query language model θ_Q with these additional words in a principled way in order to rank KB entries based on how likely they match the query entity.

3.1 Using Local Contexts

Let us look at the example from Figure 2 again. During the KB entry ranking stage, if we use θ_Q estimated from N_Q, which contains only the word "Coppola," the retrieval function is unlikely to rank the correct KB entry on the top. But if we include the contextual word "Sophia" from the query document when estimating the query language model, KL-divergence retrieval model is likely to rank the correct KB entry on the top. This idea of using contextual words to expand the query is very similar to (pseudo) relevance feedback in information retrieval. We can treat the query document D_Q as our only feedback document.

Many different (pseudo) relevance feedback methods have been proposed. Here we apply the relevance model (Lavrenko and Croft, 2001), which has been shown to be effective and robust in a recent comparative study (Lv and Zhai, 2009). We first briefly review the relevance model. Given a set of (pseudo) relevant documents $\mathcal{D}_r$, where for each $D \in \mathcal{D}_r$ there is a document language model θ_D, we can estimate a feedback language model θ_Q^{fb} as follows:

$$p(w|\theta_Q^{\text{fb}}) \propto \sum_{D \in \mathcal{D}_r} p(w|\theta_D) p(\theta_D) p(Q|\theta_D). \quad (4)$$

For our problem, since we have only a single feedback document D_Q, the equation above can be simplified. In fact, in this case the feedback language model is the same as the document language model of the only feedback document, i.e. θ_{D_Q}.

We then linearly interpolate the feedback language model with the original query language model to form an expanded query language model:

$$p(w|\theta_Q^L) = \alpha p(w|\theta_Q) + (1-\alpha) p(w|\theta_{D_Q}), \quad (5)$$

where α is a parameter between 0 and 1, to control the amount of feedback. The larger α is, the less we rely on the local context. L indicates that the query expansion comes from local context. This θ_Q^L can then replace θ_Q in Equation (1) to rank KB entries.

Special Treatment of Named Entities

Usually the document language model θ_{D_Q} is estimated using the entire text from D_Q. For entity linking, we suspect that named entities surrounding the query name string in D_Q are particularly useful for disambiguation and thus should be emphasized over other words. This can be done by weighting

NE and non-NE words differently. In the extreme case, we can use only NEs to estimate the document language model θ_{D_Q} as follows:

$$p(w|\theta_{D_Q}) = \frac{1}{K_Q}\sum_{i=1}^{K_Q}\frac{c(w, N_Q^{l,i})}{|N_Q^{l,i}|}, \quad (6)$$

where $\{N_Q^{l,i}\}$ are defined in Section 2.

Positional Model

Another observation is that words closer to the query name string in the query document are likely to be more important than words farther away. Intuitively, we can use the distance between a word and the query name string to help weigh the word. Here we apply a recently proposed positional pseudo relevance feedback method (Lv and Zhai, 2010). The document language model θ_{D_Q} now has the following form:

$$p(w|\theta_{D_Q}) = \frac{1}{K_Q}\sum_{i=1}^{K_Q} f(p_i, q)\cdot\frac{c(w, N_Q^{l,i})}{|N_Q^{l,i}|}, \quad (7)$$

where p_i and q are the absolute positions of $N_Q^{l,i}$ and N_Q in D_Q. The function f is Gaussian function defined as follows:

$$f(p, q) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\Big(\frac{-(p-q)^2}{2\sigma^2}\Big). \quad (8)$$

where variance σ controls the spread of the curve.

3.2 Using Global World Knowledge

Similar to the way we incorporate words from D_Q into the query language model, we can also construct a feedback language model using the most likely official name of the query entity obtained from Wikipedia. Specifically, we define

$$p(w|\theta_{N_Q^g}) = \frac{c(w, N_Q^g)}{|N_Q^g|}. \quad (9)$$

We can then linearly interpolate $\theta_{N_Q^g}$ with the original query language model θ_Q to form an expanded query language model θ_Q^G:

$$p(w|\theta_Q^G) = \alpha p(w|\theta_Q) + (1-\alpha)p(w|\theta_{N_Q^g}). \quad (10)$$

Here G indicates that the query expansion comes from global world knowledge.

Entity Type	%Nil	%non-Nil
GPE	32.8%	67.2 %
ORG	59.5%	40.5 %
PER	71.7%	28.3 %

Table 2: Percentages of Nil and non-Nil queries.

3.3 Combining Local Context and World Knowledge

We can further combine the two kinds of additional words into the query language model as follows:

$$p(w|\theta_Q^{L+G}) = \alpha p(w|\theta_Q) + (1-\alpha)\Big(\beta p(w|\theta_{D_Q}) + (1-\beta)p(w|\theta_{N_Q^g})\Big). \quad (11)$$

Note that here we have two parameters α and β to control the amount of contributions from the local context and from global world knowledge.

4 Experiments

4.1 Experimental Setup

Data Set: We evaluate our system on the TAC-KBP 2010 data set (Ji et al., 2010). The knowledge base was constructed from Wikipedia with 818,741 entries. The data set contains 2250 queries and query documents come from news wire and Web pages. Around 45% of the queries have non-Nil entries in the KB. Some statistics of the queries are shown in Table 2.

Tools: In our experiments, to extract named entities within D_Q and to determine T_Q, we use the Stanford NER tagger[1]. An example output of the NER tagger is shown below:

```
<PERSON>Hugh Jackman<PERSON> is Jacked!!
```

This piece of text comes from a query document where the query name string is "Jackman." We can see that the NER tagger can help locate the full name of the person.

We use the Lemur/Indri[2] search engine for retrieval. It implements the KL-divergence retrieval model as well as many other useful functionalities.

Evaluation Metric: We adopt the *Micro-averaged accuracy* metric, which is the mean accuracy over all queries. It was used in TAC-KBP 2010 (Ji et

[1] http://nlp.stanford.edu/software/CRF-NER.shtml

[2] http://www.lemurproject.org/indri.php

al., 2010) as the official metric to evaluate the performance of entity linking. This metric is simply defined as the percentage of queries that have been correctly linked.

Methods to Compare: Recall that our system consists of a KB entry selection stage and a KB entry ranking stage. At the selection stage, a set $\mathcal{S}_Q$ of alternative name strings are used to select candidate KB entries. We first define a few settings where different alternative name string sets are used to select candidate KB entries:

- **Q** represents the baseline setting which uses only the original query name string N_Q to select candidate KB entries.
- **Q+L** represents the setting where alternative name strings obtained from the query document D_Q are combined with N_Q to select candidate KB entries.
- **Q+G** represents the setting where the alternative name string obtained from Wikipedia is combined with N_Q to select candidate KB entries.
- **Q+L+G** represents the setting as we described in Section 2.1, that is, alternative name strings from both D_Q and Wikipedia are used together with N_Q to select candidate KB entries.

After selecting candidate KB entries, in the KB entry ranking stage, we have four options for the query language model and two options for the KB entry language model. For the query language model, we have (1) θ_Q, the original query language model, (2) θ_Q^L, an expanded query language model using local context from D_Q, (3) θ_Q^G, an expanded query language model using global world knowledge, and (4) θ_Q^{L+G}, an expanded query language model using both local context and global world knowledge. For the KB entry language model, we can choose whether or not to use the KB disambiguation text D_E and obtain θ_{N_E} and $\theta_{N_E+D_E}$, respectively.

4.2 Results and Discussion

First, we compare the performance of KB entry selection stage for all four settings on non-Nil queries. The performance measure recall is defined as

$$recall = \begin{cases} 1, & if\ E\ that\ refers\ to\ Q,\ exists\ in\ \mathcal{E}_Q \\ 0, & otherwise \end{cases}$$

The recall statistics in Table 3 shows that, Q+L+G has the highest recall of the KB candidate entries.

Method	Recall(%)
Q	67.1
Q+L	89.7
Q+G	94.9
Q+L+G	98.2

Table 3: Comparing the effect of candidate entry selection using different methods - KB entry selection stage recall.

Before examining the effect of query expansion in ranking, we now compare the effect of using different sets of alternative query name strings in the candidate KB entry selection stage. For this set of experiments, we fix the query language model to θ_Q and the KB entry language model to θ_{N_E} in the ranking stage.

Table 4 shows the performance of all the settings in terms of micro-averaged accuracy. The results shown in Tables 4, 5 and 6 are based on the optimum parameter settings. We can see that in terms of the overall performance, both Q+L and Q+G give better performance than Q with a 7.7% and a 9.9% relative improvement, respectively. Q+L+G gives the best performance with a 12.8% relative improvement over Q. If we further zoom into the results, we see that for ORG and PER queries, when no correct KB entry exists (i.e. the Nil case), the performance of Q, Q+L, Q+G and Q+L+G is very close, indicating that the additional alternative query name strings do not help. It shows that the alternative query name strings are most useful for queries that do have their correct entries in the KB.

We now further analyze the impact of the expanded query language models θ_Q^L, θ_Q^G and θ_Q^{L+G}. We first analyze the results without using the KB disambiguation text, i.e. using θ_{N_E}. Table 5 shows the comparison between θ_Q and other expanded query language models in terms of micro-averaged accuracy. The results reveal that the expanded query language models can indeed improve the overall performance (the both Nil and non-Nil case) under all settings. This shows the effectiveness of using the principled query expansion technique coupled with KL-divergence retrieval model to rank KB entries.

Method	All				Nil			Non-Nil		
	ALL	GPE	ORG	PER	GPE	ORG	PER	GPE	ORG	PER
Q	0.6916	0.5714	0.6533	0.8495	0.8618	**0.9888**	**0.9963**	0.4294	0.1612	0.4789
Q+L	0.7449	0.7156	0.6533	0.8655	**0.9472**	**0.9888**	0.9944	0.6024	0.1612	0.5399
Q+G	0.7604	0.7009	**0.6893**	0.8908	0.9431	**0.9888**	0.9944	0.5825	**0.2500**	0.6291
Q+L+G	**0.7800**	**0.7583**	**0.6893**	**0.8921**	0.9431	**0.9888**	0.9944	**0.6680**	**0.2500**	**0.6338**

Table 4: Comparing the performance of using different sets of query name strings for candidate KB entry selection. θ_Q and θ_{N_E} are used in KB entry ranking.

Method	QueryModel	All				Nil			Non-Nil		
		ALL	GPE	ORG	PER	GPE	ORG	PER	GPE	ORG	PER
Q+L	θ_Q	0.7449	0.7156	**0.6533**	0.8655	**0.9472**	**0.9888**	**0.9944**	0.6024	**0.1612**	0.5399
	θ_Q^L	**0.7689**	**0.7850**	**0.6533**	**0.8682**	0.9309	**0.9888**	**0.9944**	**0.7137**	**0.1612**	**0.5493**
Q+G	θ_Q	0.7604	0.7009	0.6893	0.8908	**0.9431**	**0.9888**	**0.9944**	0.5825	0.2500	0.6291
	θ_Q^G	**0.8160**	**0.7423**	**0.7867**	**0.9188**	0.9106	0.9372	0.9796	**0.6600**	**0.5658**	**0.7653**
Q+L+G	θ_Q	0.7800	0.7583	0.6893	0.8921	**0.9431**	**0.9888**	**0.9944**	0.6680	0.2500	0.6338
	θ_Q^{L+G}	**0.8516**	**0.8278**	**0.7867**	**0.9401**	0.8821	0.9372	0.9814	**0.8012**	**0.5658**	**0.8357**

Table 5: Comparison between the performance of θ_Q and expanded query language models in terms of micro average accuracy. θ_{N_E} was used in ranking.

On the other hand, again we observe that the effects on the Nil and the non-Nil queries are different. While in Table 4 the alternative name strings do not affect the performance much for Nil queries, now the expanded query language models actually hurt the performance for Nil queries. It is not surprising to see this result. When we expand the query language model, we can possibly introduce noise, especially when we use the external knowledge obtained from Wikipedia, which largely depends on what Wikipedia considers to be the most popular official name of a query name string. With noisy terms in the expanded query language model we increase the chance to link the query to a KB entry which is not the correct match. The challenge is that we do not know when additional terms in the expanded query language model are noise and when they are not, because for non-Nil queries we do observe a substantial amount of improvement brought by query expansion, especially with external world knowledge. We will further investigate this research question in the future.

We now further study the impact of using the KB disambiguation text associated with each entry to estimate the KB entry language model used in the KL-divergence ranking function. The results are shown in Table 6 for all the methods on θ_{N_E} vs. $\theta_{N_E+D_E}$ using the expanded query language models. We can see that for all methods the impact of using the KB disambiguation text is very minimal and is observed only for GPE and ORG queries. Table 7 shows an example of the KL-divergence scores for a query, *Mobile* whose context is previously shown in the Figure 1. Without the KB disambiguation text both the KB entry *Mobile Alabama* and the entry *Mobile River* are given the same score, resulting in inaccurate linking in the θ_{N_E} case. But with $\theta_{N_E+D_E}$, *Mobile Alabama* was scored higher, resulting in an accurate linking. However, we observe that such cases are very rare in the TAC 2010 query list and thus the overall improvement observed is minimal.

KB Entry	KB Name	w/o text	w/ text
E0583976	Mobile Alabama	-6.28514	-6.3839
E0183287	Mobile River	-6.28514	-6.69372

Table 7: The KL-divergence scores of KB entities for the query *Mobile*.

Finally, we compare our performance with the highest scores from TAC-KBP 2010 as shown in the Table 8. It is important to note that the highest TAC results shown in the table under each setting are not necessarily obtained by the same team. We can see that our overall performance when KB text is used is competitive compared with the highest TAC score, and is substantially higher than the TAC score when KB text is not used. Lehmann et al. (2010) achieved highest TAC scores. They used a variety of evidence from Wikipedia like disambiguation pages, anchors, expanded acronyms and redirects to build a rich feature set. But as we discussed, building a rich fea-

Method	KB Text	All				Nil			Non-Nil		
		ALL	GPE	ORG	PER	GPE	ORG	PER	GPE	ORG	PER
Q	θ_{N_E}	**0.6916**	**0.5714**	0.6533	0.8495	0.8618	0.9888	0.9963	0.4294	0.1612	0.4789
	$\theta_{N_E+D_E}$	0.6888	0.5607	0.6533	0.8495	0.8618	0.9888	0.9963	0.4135	0.1612	0.4789
Q+L	θ_{N_E}	0.7689	0.7850	0.6533	0.8682	0.9309	0.9888	0.9944	0.7137	0.1612	0.5493
	$\theta_{N_E+D_E}$	**0.7707**	**0.7904**	0.6533	0.8682	**0.9390**	0.9888	0.9944	**0.7177**	0.1612	0.5493
Q+G	θ_{N_E}	0.8160	0.7423	**0.7867**	0.9188	**0.9106**	0.9372	0.9796	0.6600	**0.5658**	0.7653
	$\theta_{N_E+D_E}$	**0.8222**	**0.7450**	0.7827	**0.9387**	0.8902	0.9372	**0.9814**	**0.6740**	0.5559	**0.8310**
Q+L+G	θ_{N_E}	0.8516	0.8278	0.7867	0.9401	**0.8821**	0.9372	0.9814	0.8012	0.5658	0.8357
	$\theta_{N_E+D_E}$	**0.8524**	**0.8291**	**0.7880**	0.9401	0.8740	0.9372	0.9814	**0.8072**	**0.5691**	0.8357

Table 6: Comparing the performance using KB text and without using KB text for all methods using expanded query models in terms of micro average accuracy on 2250 queries. $\theta_{N_E+D_E}$ represents method using KB text and θ_{N_E} represents methods without using KB text.

ture set is an expensive task. Their overall accuracy is 1.5% higher than our model. Table 8 shows that the performance of ORG entities is lower when compared with the TAC results when we used KB text. In our analysis, we observed that, even though some entities like AMPAS are linked correctly, the entities like CCC (Consolidated Contractors Company) failed due to ambiguity in the title. Here, we may benefit by leveraging more global knowledge, i.e, we should expand the N_Q^g with Wikipedia global context entities together with the title to fully benefit from global knowledge. In particular, when KB text is not used, our system outperforms the highest TAC results for all three types of queries.

From the analysis by Ji et al. (2010), overall the participating teams generally performed the best on PER queries and the worst on GPE queries. With our system, we can achieve good performance on GPE queries.

KB Text Usage	Type	Our System	TAC Highest
$\theta_{N_E+D_E}$	All	0.8524	**0.8680**
	GPE	**0.8291**	0.7957
	ORG	0.7880	**0.8520**
	PER	0.9401	**0.9601**
θ_{N_E}	All	**0.8516**	0.7791
	GPE	**0.8278**	0.7076
	ORG	**0.7867**	0.7333
	PER	**0.9401**	0.9001

Table 8: Comparison of the best configuration of our system (Q+L+G with θ_Q^{L+G}) with the TAC-KBP 2010 results in terms of micro-averaged accuracy. $\theta_{N_E+D_E}$ represents the method using KB disambiguation text and θ_{N_E} represents the method without using KB disambiguation text.

4.3 Parameter Sensitivity

In all our experiments, we set the Dirichlet prior μ to 2500 following previous studies. For the threshold τ we empirically set it to -12.0 in all the experiments based on preliminary results. Recall that all the expanded query language models also have a control parameters α. The local context-based models θ_Q^L and θ_Q^{L+G} have an additional parameter σ which controls the proximity weighing. The θ_Q^{L+G} model has another additional parameter β that controls the balance between local context and world knowledge. In this subsection, we study the sensitivity of these parameters. We plot the sensitivity graphs for all the methods that involve α (β set to 0.5) in Figure 3. As we can see, all the curves appear to be stable and α=0.4 appears to work well.

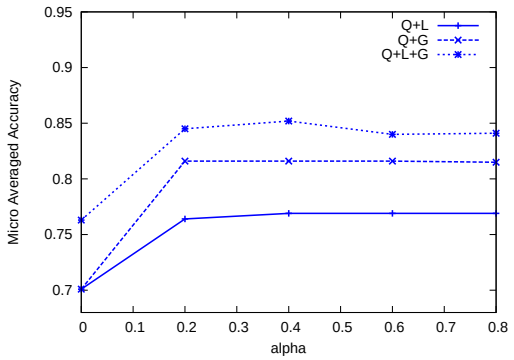


Figure 3: Sensitivity of α in regard to micro-averaged accuracy.

Similarly, we set α=0.4 and examine how β affects micro averaged accuracy. We plot the sensitivity curve for β for the Q+L+G setting with θ_Q^{L+G} in Figure 5. As we can see, the best performance is achieved when β=0.5. This implies that the local

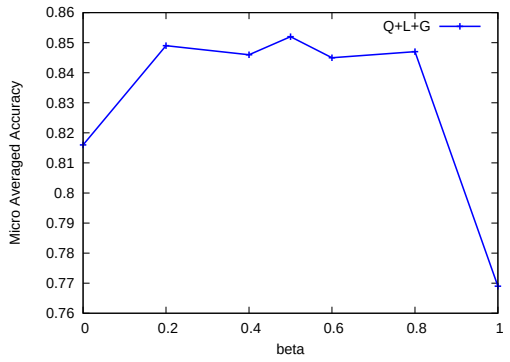


Figure 4: Sensitivity of β in regard to micro-averaged accuracy.

context and the global world knowledge are weighed equally for aiding disambiguation and improving the entity linking performance.

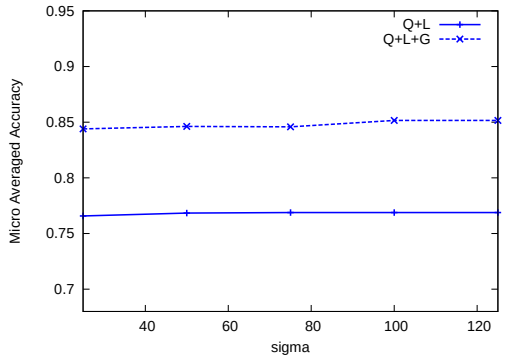


Figure 5: Sensitivity of σ with respect to micro-averaged accuracy.

Furthermore, we systematically test a fixed set of σ values from 25 to 125 with an intervals of 25 and examine how σ affects micro averaged accuracy. We set α=0.4 and β=0.5, which is the best parameter setting as discussed above. We plot the sensitivity curves for the parameter σ for methods that utilize the local context, i.e. θ_Q^L and θ_Q^{L+G}, in Figure 5. We observe that all the curves are stable and 75 $<=$ σ $<=$ 100 appears to work well. We set σ=100 for all our experiments. Moreover, after 100, the graph becomes stable, which indicates that proximity has less impact on the method from this point on. This implies that an equal weighing scheme actually would work the same for these experiments. Part of the reason may be that by using only named entities in the context rather than all words, we have effectively picked the most useful contextual terms. Therefore, positional feedback models do have exhibit much benefit for our problem.

5 Related Work

Bunescu and Pasca (2006) and Cucerzan (2007) explored the entity linking task using Vector Space Models for ranking. They took a classification approach together with the novel idea of exploiting Wikipedia knowledge. In their pioneering work, they used Wikipedia's category information for entity disambiguation. They show that using different background knowledge, we can find efficient approaches for disambiguation. In their work, they took an assumption that every entity has a KB entry and thus the NIL entries are not handled.

Similar to other researchers, Zhang et al. (2010) took an approach of classification and used a two-stage approach for entity liking. They proposed a supervised model with SVM ranking to filter out the candidates and deal with disambiguation effectively. For entity diambiguation they used the contextual comparisons between the Wikipedia article and the KB article. However, their work ignores the possibilities of acronyms in the entities. Also, the ambiguous geo-political names are not handled in their work.

Dredze et al. (2010) took the approach that large number of entities will be unlinkable, as there is a probability that the relevant KB entry is unavailable. Their algorithm for learning NIL has shown very good results. But their proposal for handling the alias name or stage name via multiple lists is not scalable. Unlike their approach, we use the global knowledge to handle the stage names and thus this gives an optimized solution to handle alias names. Similarly, for acronyms we use the global knowledge that aids unabbreviating and thus entity disambiguation. Similar to other approaches, Zheng et al. (2010) took a learning to rank approach and compared list-wise rank model to the pair-wise rank model. They achieved good results on the list-wise ranking approach. They handled acronyms and disambiguity through wiki redirect pages and the anchor texts which is similar to others ideas.

Challenges in supervised learning includes careful feature selection. The features can be selected in ad hoc manner - similarity based or semantic based. Also machine learning approach induces challenges of handling heterogenous cases. Unlike their machine learning approach which requires careful fea-

ture engineering and heterogenous training data, our method is simple as we use simple similarity measures. At the same time, we propose a statistical language modeling approach to the linking problem. Many researchers have proposed efficient ideas in their works. We integrated some of their ideas like world knowledge with our new techniques to achieve efficient entity linking accuracy.

6 Conclusions

In this paper we proposed a novel approach to entity linking based on statistical language model-based information retrieval with query expansion using the local context from the query document as well as world knowledge from the Web. Our model is a simple unsupervised one that follows principled existing information retrieval techniques. And yet it performs the entity linking task effectively compared with the best performance achieved in the TAC-KBP 2010 evaluation.

Currently our model does not exploit world knowledge from the Web completely. World knowledge, especially obtained from Wikipedia, has shown to be useful in previous studies. As our future work, we plan to explore how to further incorporate such world knowledge into our model in a principled way.

References

Razvan Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 9–16, Trento, Italy.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 708–716.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 277–285.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the TAC 2010 knowledge base population track. In *Proceedings of the Third Text Analysis Conference*.

John Lafferty and ChengXiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111–119.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127.

John Lehmann, Sean Monahan, Luke Nezda, Arnold Jung, and Ying Shi. 2010. Lcc approaches to knowledge base population at tac 2010. In *Proceedings TAC 2010 Workshop*. TAC 2010.

Yuanhua Lv and ChengXiang Zhai. 2009. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 1895–1898.

Yuanhua Lv and ChengXiang Zhai. 2010. Positional relevance model for pseudo-relevance feedback. In *Proceeding of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 579–586.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the TAC 2009 knowledge base population track. In *Proceedings of the Second Text Analysis Conference*.

ChengXiang Zhai and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 403–410.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, April.

Wei Zhang, Jian Su, Chew Lim Tan, and Wen Ting Wang. 2010. Entity linking leveraging automatically generated annotation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1290–1298.

Zhicheng Zheng, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 483–491.

Unsupervised Information Extraction with Distributional Prior Knowledge

Cane Wing-ki Leung[1], **Jing Jiang**[1], **Kian Ming A. Chai**[2], **Hai Leong Chieu**[2], **Loo-Nin Teow**[2]
[1]School of Information Systems, Singapore Management University, Singapore
[2]DSO National Laboratories, Singapore
{caneleung,jingjiang}@smu.edu.sg, {ckianmin,chaileon,tloonin}@dso.org.sg

Abstract

We address the task of automatic discovery of information extraction template from a given text collection. Our approach clusters candidate slot fillers to identify meaningful template slots. We propose a generative model that incorporates distributional prior knowledge to help distribute candidates in a document into appropriate slots. Empirical results suggest that the proposed prior can bring substantial improvements to our task as compared to a K-means baseline and a Gaussian mixture model baseline. Specifically, the proposed prior has shown to be effective when coupled with discriminative features of the candidates.

1 Introduction

Information extraction (IE) is the task of extracting information from natural language texts to fill a database record following a structure called a *template*. Such templates are usually defined based on the domain of interest. For example, the domain in the Sixth Message Understanding Conference (MUC-6, 1995) is management succession, and the pre-defined template consists of the slots *position*, the *person leaving*, the *person joining*, and the *organization*.

Previous research on IE often requires the predefinition of templates. Template construction is usually done manually by domain experts, and annotated documents are often created to facilitate supervised learning approaches to IE. However, both manual template construction and data annotation are labor-intensive. More importantly, templates and annotated data usually cannot be re-used in new domains due to domain dependency. It is therefore natural to consider the problem of unsupervised template induction and information extraction. This is the topic of this paper.

There have been a few previous attempts to address the unsupervised IE problem (Shinyama and Sekine, 2006; Sekine, 2006; Rosenfeld and Feldman, 2006; Filatova et al., 2006). These approaches have a commonality: they try to cluster candidate slot fillers, which are often nouns and noun phrases, into slots of the template to be constructed. However, most of them have neglected the following important observation: a single document or text segment tends to cover different slots rather than redundantly fill the same slot. In other words, during clustering, candidates within the same text segment should be more likely to be distributed into different clusters.

In this paper, we propose a generative model that incorporates this distributional prior knowledge. We define a prior distribution over the possible label assignments in a document or a text segment such that a more diversified label assignment is preferred. This prior is based on the Poisson distribution. We also compare a number of generative models for generating slot fillers and find that the Gaussian mixture model is the best. We then combine the Poisson-based label assignment prior with the Gaussian mixture model to perform slot clustering. We find that compared with a K-means baseline and a Gaussian mixture model baseline, our combined model with the proposed label assignment prior substantially performs better on two of the three data sets we use for evaluation. We further analyze the results on the third data set and find that the proposed prior will have little effect if there are no good discriminative features to begin with. In summary, we find that

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 814–824,
Edinburgh, Scotland, UK, July 27–31, 2011.

our Poisson-based label assignment prior is effective when coupled with good discriminative features.

2 Related Work

One common approach to unsupervised IE is based on automatic IE pattern acquisition on a cluster of similar documents. For instance, Sudo et al. (2003) and Sekine (2006) proposed different methods for automatic IE pattern acquisition for a given domain based on frequent subtree discovery in dependency parse trees. These methods leveraged heavily on the entity types of candidates when assigning them to template slots. As a consequence, potentially different semantic roles of candidates having the same entity type could become indistinguishable (Sudo et al., 2003; Sekine, 2006). This problem is alleviated in our work by exploiting distributional prior knowledge about template slots, which is shown effective when coupled with discriminative features of candidates. Filatova et al. (2006) also considered frequent subtrees in dependency parse trees, but their goal was to build templates around verbs that are statistically important in a given domain. Our work, in contrast, is not constrained to verb-centric templates. We aim to identify salient slots in the given domain by clustering.

Marx et al. (2002) proposed the cross-component clustering algorithm for unsupervised IE. Their algorithm assigned a candidate from a document to a cluster based on the candidate's feature similarity with candidates from *other documents only*. In other words, the algorithm did not consider a candidate's relationships with other candidates in the same document. Our work is based on a different perspective: we model label assignments for all candidates in the same document with a distributional prior that prefers a document to cover more distinct slots. We show empirically that this prior improves slot clustering results greatly in some cases.

Also related to our work is open domain IE, which aims to perform unsupervised *relation extraction*. TEXTRUNNER (Banko et al., 2007), for example, automatically extracts *all possible relations* between pairs of noun phrases from a given corpus. The main difference between open domain IE and our work is that open domain IE does not aim to induce domain templates, whereas we focus on a single domain with the goal of inducing a template that describes salient information structure of that domain. Furthermore, TEXTRUNNER and related studies on unsupervised relation extraction often rely on highly redundant information on the Web or in large corpus (Hasegawa et al., 2004; Rosenfeld and Feldman, 2006; Yan et al., 2009), which is not assumed in our study.

We propose a generative model with a distributional prior for the unsupervised IE task, where slot fillers correspond to *observations* in the model, and their labels correspond to *hidden variables* we want to learn. In the machine learning literature, researchers have explored the use of similar prior knowledge in the form of *constraints* through model expectation. For example, Graça et al. (2007) proposed to place constraints on the posterior probabilities of hidden variables in a generative model, while Druck et al. (2008) studied a similar problem in a discriminative, semi-supervised setting. These studies model constraints as features, and enforce the constraints through expected feature values. In contrast, we place constraints on label assignments through a probabilistic prior on the distribution of slots. The proposed prior is simple and easy to interpret in a generative model. Nevertheless, it will be interesting to explore how the proposed prior can be implemented within the posterior constraint framework.

3 Problem Overview

In this section, we first formally define our unsupervised IE problem. We then provide an overview of our solution, which is based on a generative model.

3.1 Problem Definition

We assume a collection of documents or short text segments from a certain domain. These documents or text segments describe different events or entities, but they are about the same topic or aspect of the domain. Examples of such collections include a collection of sentences describing the educational background of famous scientists and a collection of aviation incident reports. Our task is to automatically discover an IE template from this collection. The discovered template should contain a set of slots that play different semantic roles in the domain.

Input text:

Topic: Graduate Student Seminar Lunch
Dates: 13-Apr-95
Time: 12:00 PM - 1:30 PM
PostedBy: Edmund J. Delaney on 5-Apr-95 at 16:24 from andrew.cmu.edu
Abstract:
The last Graduate Student Seminar Series lunch will be held on Thursday, April 13 from noon-1:30 p.m. in room 207, Student Activities Center. Professor Sara Kiesler of SDS will speak on Carving A Successful Research Niche.

Output:

Slot	Slot Filler(s)
Slot 1 (*start time*)	12:00PM
Slot 2 (*end time*)	1:30PM, 1:30 p.m.
Slot 3 (*location*)	room 207, Student Activities Center
Slot 4 (*speaker*)	Professor Sara Kiesler
Slot 4 (*irrelevant information*)	Edmund J. Delaney

Figure 1: An input text from a seminar announcement collection and the discovered IE template. Note that the slots are automatically discovered and the slot names are manually assigned.

To construct such a template, we start with identifying candidate slot fillers, hereafter referred to as *candidates*, from the input text. Then we cluster these candidates with the aim that each cluster will represent a semantically meaningful slot. Figure 1 gives an example of an input text from a collection of seminar announcements and the resulting template discovered from the collection. As we can see, the template contains some semantically meaningful slots such as the *start time*, *end time*, *location* and *speaker* of a seminar. Moreover, it also contains a slot that covers an irrelevant candidate. We call such slots covering irrelevant candidates *garbage slots*.

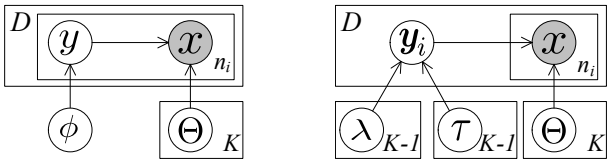
We can make two observations on the mapping from candidates to template slots from real data, such as the text in Figure 1. Firstly, a template slot may be filled by more than one candidate from a single document, although this number has been observed to be small. For example, the template slot *end time* in Figure 1 has two slot fillers: "1:30 PM" from the semi-structured header and "1:30 p.m." from within the abstract. Secondly, a document tends to contain candidates that cover different template slots. We believe that this observation is a consequence of the fact that a document will tend to convey as much information as possible. We further exploit these observations in Section 4.

3.2 A General Solution

Recall that our general solution to the unsupervised IE problem is to cluster candidate slot fillers in order to identify meaningful slots. We leave the details of how to extract the candidates to Section 7.1. In this section, we assume that we have a set of candidates $\boldsymbol{x} = \{x_{i,j}\}$, where $x_{i,j}$ is the j-th candidate from the i-th document in the collection. We cluster these candidates into K groups for a given K.

Let $y_{i,j} \in \{1, \ldots, K\}$ denote the cluster label for $x_{i,j}$ and $\boldsymbol{y}$ denote the set of all the $y_{i,j}$'s. Let $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ denote the sets of all the $x_{i,j}$'s and the $y_{i,j}$'s in the i-th document respectively. We assume a generative model for $\boldsymbol{x}$ and $\boldsymbol{y}$ as follows. For the i-th document in our collection, we assume that the number of candidates is known and we draw a label assignment $\boldsymbol{y}_i$ according to some distribution parameterized by Λ. Then for the j-th candidate, we generate $x_{i,j}$ from $y_{i,j}$ according to a generative model parameterized by Θ. Since the labels $\boldsymbol{y}$ are hidden, the observed log-likelihood of the parameters given the observations $\boldsymbol{x}$ is

$$\begin{aligned} L(\Lambda, \Theta) &= \log p(\boldsymbol{x}; \Lambda, \Theta) \\ &= \sum_i \log \sum_{\boldsymbol{y}_i} p(\boldsymbol{x}_i, \boldsymbol{y}_i; \Lambda, \Theta) \\ &= \sum_i \log \sum_{\boldsymbol{y}_i} p(\boldsymbol{y}_i; \Lambda) \prod_j p(x_{i,j} | y_{i,j}; \Theta). \quad (1) \end{aligned}$$



(a) A multinomial prior. (b) The proposed Poisson-based prior.

Figure 2: Generative models with different label assignment priors. D denotes the number of documents in the given collection, n_i denotes the number of candidates in the i-th document, and K is the number of slots (clusters).

For a given functional form of $p(\boldsymbol{y}_i; \Lambda)$ and $p(x_{i,j}|y_{i,j}; \Theta)$, the best model parameters can be estimated by maximizing Eq. (1). In the next section, we detail two designs of the prior $p(\boldsymbol{y}_i; \Lambda)$, followed by different generative models for the distribution $p(x_{i,j}|y_{i,j}; \Theta)$ in Section 5. Then we describe the estimation of model parameters in Section 6.

4 Label Assignment Prior

The label assignment prior, $p(\boldsymbol{y}_i; \Lambda)$, models the generation of labels for candidates in a document. In this section, we first describe a commonly used multinomial prior, and then introduce the proposed Poisson-based prior for the unsupervised IE task.

4.1 A Multinomial Prior

Usually, one would assume that the labels for the different candidates in the same document are generated independently, that is, $p(\boldsymbol{y}_i; \Lambda) = \prod_j p(y_{i,j}; \Lambda)$. Under this model, we assume that each $y_{i,j}$ is generated from a multinomial distribution parameterized by $\boldsymbol{\phi}$, where ϕ_y denotes the probability of generating label y. Our objective function in Eq. (1) then becomes:

$$\begin{aligned} L(\Lambda, \Theta) &= \log p(\boldsymbol{x}; \Lambda, \Theta) \\ &= \sum_{i,j} \log \sum_y \phi_y p(x_{i,j}|y; \Theta). \end{aligned} \quad (2)$$

Figure 2(a) depicts a generative model with this multinomial prior in plate notation. Note that the independence assumption on label assignment in this model does not capture our observation that candidates in a document are likely to cover different semantic roles.

4.2 The Proposed Poisson-based Prior

We propose a prior distribution that favors more diverse label assignments. Our proposal takes into consideration the following three observations. Firstly, candidates in the same document are likely to cover different semantic roles. The proposed prior distribution should therefore assign higher probability to a label assignment that covers more distinct slots. Secondly, the same piece of information is not likely to be repeated many times in a document. Our design thus allows a slot to generate multiple fillers in a document, up to a limited number of times. Thirdly, there may exist candidates that do not belong to slots in the extracted template. Therefore, we introduce a *dummy slot* or *garbage slot* to the label set to collect such candidates. Yet, we shall not assume any prior/domain knowledge about candidates generated by the garbage slot as they are essentially irrelevant in the given domain.

We now detail the prior that exploits the above observations. First, we fix the K-th slot (or cluster) in the label set to be the garbage slot. For each of the non-garbage slot $k = 1, \ldots, K-1$, we also fix the maximum number of fillers that can be generated, which we denote by λ_k. There is no λ_K for the garbage slot because the number of fillers is not constrained for this slot. This allows all candidates in a document to be generated by the garbage slot. Let n_i be the number of candidates in the i-th document. Given K, $\{\lambda_k\}_{k=1}^{K-1}$ and n_i, the set of possible label assignments for the i-th document can be generated. We illustrate this with an example. Let $K = 2$ and $\lambda_1 = 1$. The label set is $\{1, 2\}$, where 2 represents the garbage slot. Let the number of candidates be $n_i = 2$. The possible label assignments within this setting are $(1, 2)$, $(2, 1)$ and $(2, 2)$.

The set of possible label assignments for the i-th document is the sample space on which we place the prior distribution $p(\boldsymbol{y}_i; \Lambda)$. We need a prior that gives a higher probability to a more diverse label assignment. For a given $\boldsymbol{y}_i$ for the i-th document, let $n_{i,k}$ be the number of candidates in the document that have been assigned to slot k. That is, $n_{i,k} \stackrel{\text{def}}{=} \sum_{j=1}^{n_i} \mathbf{1}(y_{i,j} = k)$, where $\mathbf{1}(\cdot)$ is the indicator variable. We propose the following distribution based on the Poisson distribution:

$$p(\boldsymbol{y}_i; \Lambda) \stackrel{\text{def}}{=} Z_i^{-1} \prod_{k=1}^{K-1} \text{Poisson}(n_{i,k}; \tau_k), \quad (3)$$

where Z_i is the normalizing constant, and τ_k is the mean parameter of the k-th Poisson distribution, $k = 1, \ldots K-1$. The absence of a factor that depends on $n_{i,K}$ reflects the lack of prior knowledge on the number of garbage slot fillers. Figure 2(b) depicts the proposed generative model with the Poisson-based prior in plate notation.

5 Generating Slot Fillers

Different existing generative models can be used to model the generation of a slot filler given a label, that is, $p(x|y;\Theta)$. We explore four of them for our task, namely, the naive Bayes model, the Bernoulli mixture model, the Gaussian mixture model, and a locally normalized logistic regression model proposed by Berg-Kirkpatrick et al. (2010).

5.1 Multinomial Naive Bayes

In the multinomial naive Bayes model, features of an observation x are assumed to be independent and each generated from a multinomial distribution. We first introduce some notations. Let f denote a feature (e.g. entity type) and $\mathcal{V}_f$ denote the set of possible values for f. Let $x^f \in \mathcal{V}_f$ be the value of feature f in x (e.g. *person*). For a given label y, feature f follows a multinomial distribution parameterized by $\boldsymbol{\psi}_{y,f}$, where $\psi_{y,f,v}$ denotes the probability of feature f taking the value $v \in \mathcal{V}_f$ given label y. The functional form of the conditional probability of x given a label y is then

$$p(x|y;\Theta) = \prod_f p(x^f|y;\Theta) = \prod_f \psi_{y,f,x^f}. \quad (4)$$

5.2 Bernoulli Mixture Model

In the naive Bayes model our features are defined to be categorical. For the Bernoulli mixture model, as well as the Gaussian mixture model and the locally normalized logistic regression model in the next subsections, we first convert each observation x into a binary feature vector $\mathbf{x} \in \{0,1\}^F$ where F is the number of binary features. An example of a binary features is "the entity type is *person*".

We assume that, for a given label y, observations are generated from a multivariate Bernoulli distribution parameterized by $\boldsymbol{\varphi}_{y,f}$, where $\varphi_{y,f,v}$ denotes the probability of feature f taking the value $v \in \{0,1\}$ given label y. The conditional probability of $\mathbf{x}$ given y can then be written as

$$\begin{aligned} p(\mathbf{x}|y;\Theta) &= \prod_f p(x_f = 1|y;\Theta)^{x_f} \cdot p(x_f = 0|y;\Theta)^{1-x_f} \\ &= \prod_f \varphi_{y,f,x^f}. \end{aligned} \quad (5)$$

5.3 Gaussian Mixture Model

In the Gaussian mixture model, we assume that a given label y generates observations with a multivariate Gaussian distribution $\mathcal{N}(\mu_y, \Sigma_y)$, where $\mu_y \in \mathbb{R}^F$ is the mean and $\Sigma_y \in \mathbb{R}^{F \times F}$ is the covariance matrix of the Gaussian. If we assume that the different feature dimensions are independent and have the same variance, that is, $\Sigma_y = \sigma_y^2 I$, where I is the identity matrix, then the conditional density of $\mathbf{x}$ given y is

$$p(\mathbf{x}|y;\Theta) = \frac{1}{(2\pi\sigma_y^2)^{F/2}} \exp\left(-\frac{\|\mathbf{x} - \mu_y\|^2}{2\sigma_y^2}\right). \quad (6)$$

5.4 Locally Normalized Logistic Regression

Berg-Kirkpatrick et al. (2010) proposed a method for incorporating features into generative models for unsupervised learning. Their method models the generation of $\mathbf{x}$ given y as a logistic function parameterized by a weight vector $\mathbf{w}_y$, defined as follows:

$$p(\mathbf{x}|y;\Theta) = \frac{\exp\langle \mathbf{x}, \mathbf{w}_y \rangle}{\sum_{\mathbf{x}'} \exp\langle \mathbf{x}', \mathbf{w}_y \rangle}. \quad (7)$$

$\langle \mathbf{x}, \mathbf{w} \rangle$ denotes the inner product between $\mathbf{x}$ and $\mathbf{w}$. The denominator considers all data points $\mathbf{x}'$ in the data set, thus Eq. (7) gives a probability distribution over data points for a given y.

6 Parameter Estimation

We can apply the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) to maximize the log-likelihood functions under both multinomial prior in Eq. (2) and the proposed Poisson-based prior in Eq. (1). For the multinomial prior, there are standard closed form solutions for the naive Bayes, the Bernoulli mixture and the Gaussian mixture models. For locally normalized logistic regression, model parameters can also be learned via EM, but with a gradient-based M-step (Berg-Kirkpatrick et al., 2010). We leave out the details here and focus on parameter estimation in the proposed generative model with the Poisson-based prior.

We assume that in the Poisson-based prior, the parameters $\{\lambda_k\}_{k=1}^{K-1}$ and $\{\tau_k\}_{k=1}^{K-1}$ are fixed rather than learned in this work. For the distribution $p(x|y;\Theta)$, let $\Theta^{(t-1)}$ and $\Theta^{(t)}$ denote parameter estimates from two consecutive EM iterations. At the t-th iteration, the E-step updates the responsibilities of each label assignment $\boldsymbol{y}_i$ for each document:

$$\begin{aligned}\alpha_{i,\boldsymbol{y}_i} &= p(\boldsymbol{y}_i|\boldsymbol{x_i};\Lambda,\Theta^{(t-1)}) \\ &= \frac{p(\boldsymbol{y}_i;\Lambda)p(\boldsymbol{x}_i|\boldsymbol{y}_i;\Theta^{(t-1)})}{\sum_{\boldsymbol{y}'_i} p(\boldsymbol{y}'_i;\Lambda)p(\boldsymbol{x}_i|\boldsymbol{y}'_i;\Theta^{(t-1)})}, \quad (8)\end{aligned}$$

where α_i is a distribution over all possible label assignments $\boldsymbol{y}_i$'s for the i-th document. The M-step updates the estimates of $\Theta^{(t)}$ based on the current values of α_i's and $\Theta^{(t-1)}$. This is done by maximizing the following objective function:

$$\sum_i \sum_{\boldsymbol{y}_i} \alpha_{i,\boldsymbol{y}_i} \log\Big(p(\boldsymbol{y}_i;\Lambda)\prod_j p(x_{i,j}|y_{i,j};\Theta^{(t-1)})\Big). \quad (9)$$

The exact formulas used in the M-step for updating Θ depend on the functional form of $p(x_{i,j}|y_{i,j};\Theta)$. As an example, we give the formulas for the Gaussian mixture model, in which Θ contains the set of means $\{\mu_k^{(t)}\}_{k=1}^K$ and variances $\{\sigma_k^{(t)}\}_{k=1}^K$. Taking the derivatives of Eq. (9) with respect to μ_k and to σ_k, and then setting the derivations to zero, we can solve for μ_k and for σ_k to get:

$$\mu_k^{(t)} = \frac{\sum_i \sum_{\boldsymbol{y}_i} \alpha_{i,\boldsymbol{y}_i} \sum_j \mathbf{1}(y_{i,j}=k)\mathbf{x}_{i,j}}{\sum_i \sum_{\boldsymbol{y}_i} \alpha_{i,\boldsymbol{y}_i} \sum_j \mathbf{1}(y_{i,j}=k)}, \quad (10)$$

$$\sigma_k^{(t)} = \frac{\sum_i \sum_{\boldsymbol{y}_i} \alpha_{i,\boldsymbol{y}_i} \sum_j \mathbf{1}(y_{i,j}=k)||\mathbf{x}_{i,j}-\mu_k^{(t)}||^2}{F\sum_i \sum_{\boldsymbol{y}_i} \alpha_{i,\boldsymbol{y}_i} \sum_j \mathbf{1}(y_{i,j}=k)}, \quad (11)$$

where $\mathbf{1}(\cdot)$ is the indicator variable. We skip the derivations here due to space limit.

Closed form solutions also exist for the naive Bayes and the Bernoulli mixture models. For locally normalized logistic regression, parameters can be learned with a gradient-based M-step as in the multinomial prior setting. Existing optimization algorithms, such as L-BFGS, can be used for optimizing model parameters in the M-step as discussed in (Berg-Kirkpatrick et al., 2010).

7 Experiments

In this section, we first describe the data sets we used in our experiments, detailing the target slots and candidates in each data set, as well as features we extract for the candidates. We then describe our evaluation metrics, followed by experimental results.

7.1 Data Sets

We use three data sets for evaluating our unsupervised IE task. Note that to speed up computation, we only include documents or text segments containing no more than 10 candidates in our experiments. The first data set contains a set of seminar announcements (Freitag and McCallum, 1999), annotated with four slot labels, namely *stime* (start time), *etime* (end time), *speaker* and *location*. We used as candidates all strings labeled in the annotated data as well as all named entities found by the Stanford NER tagger for CoNLL (Finkel et al., 2005). There are 309 seminar announcements with 2262 candidates in this data set.

The second data set is a collection of paragraphs describing aviation incidents, taken from the Wikipedia article on "List of accidents and incidents involving commercial aircraft" (Wikipedia, 2009). Each paragraph in the article contains one to a few sentences describing an incident. In this domain, we take each paragraph as a separate document, and all hyperlinked phrases in the original Wikipedia article as candidates. For evaluation, we manually annotated the paragraphs of incidents from 2006 to 2009 with five slot labels: the *flight number* (FN), the *airline* (AL), the *aircraft model* (AC), the *exact*

location (LO) of the incident (e.g. airport name), and the *country* (CO) where the incident occurred. The entire data set consists of 564 paragraphs with 2783 candidates. The annotated portion consists of 74 paragraphs with 395 candidates.

The third data set comes from the management succession domain used in the Sixth Message Understanding Conference (MUC-6, 1995). We extract from the original data set all sentences that were tagged with a management succession event, and use as candidates all tagged strings in those sentences. This domain has four target slots, namely *PersonIn* (the person moving into a new position), *PersonOut* (the person leaving a position), *Org* (the corporation's name) and *Post* (the position title). Sentences containing candidates with multiple labels (candidates annotated as both *PersonIn* and *PersonOut*) are discarded. The extracted data set consists of 757 sentences with 2288 candidates.

7.2 Features

To extract features for candidates, we first normalize each word to its lower-case, with digits replaced by the token *digit*. We extract the following features for every candidate: the candidate phrase itself, its head word, the unigram and bigram before and after the candidate in the sentence where it appeared, its entity type (*person*, *location*, *organization*, and *date/time*), as well as features derived from dependency parse trees. Specifically, we first apply the Stanford lexical parser to our data (de Marneffe et al., 2006). Then for each candidate, we follow its dependencies in the corresponding dependency parse tree until we find a relation $r \in \{nsubj, csubj, dobj, iobj, pobj\}$ in which the candidate is the dependent. We then construct a feature (r, v) where v is governor of the relation.

7.3 Evaluation Baseline and Method

We use the standard K-means algorithm (Macqueen, 1967) as a non-generative baseline, since K-means is commonly used for clustering. To evaluate clustering results, we match each slot in the labeled data to the cluster that gives the best F1-measure when evaluated for the slot. We report the precision (P), recall (R) and F1-measure for individual slot labels, as well as the macro- and micro- average results across all labels for each experiment. We conduct 10 trials of experiment on each model and each data set with different random initializations. We report the trials that give the smallest within-cluster sum-of-squares (WCSS) distance for K-means, and those that give the highest log-likelihood of data for all other models. Experimental trials are run until the change in WCSS/log-likelihood between two EM iterations is smaller than 1×10^{-6}. All trials converged within 30 minutes.

All models we evaluate involve a parameter K, which is the number of values that y can take on. The value of K is manually fixed in this study. As noted, we use a garbage slot to capture irrelevant candidates, thus the value of K is set to the number of target slots plus 1 for each data set. We empirically set the adjustable parameters in the proposed prior, and the weight of the regularization term in the locally normalized logistic regression model (Berg-Kirkpatrick et al., 2010), denoted by β. Exact settings are given in the next subsection. Note that the focus of our experiments is on evaluating the effectiveness of the proposed prior. We leave the task of learning the various parameter values to future work.

7.4 Results

Evaluation on existing generative models

We first evaluate the existing generative models described in Section 5 with the multinomial prior. Table 1 summarizes the performance of Naive Bayes (NB), the Bernoulli mixture model (BMM), the Gaussian mixture model (GMM), the locally normalized logistic regression (LNLR) model, and K-means. We only show the F1 measures in the table due to space limit.

We first observe that NB does not perform well for our task. LNLR, which is an interesting contribution in its own right, does not seem to be suitable for our task as well. While NB and LNLR are inferior to K-means for all three data sets, BMM shows mixed results. Specifically, BMM outperforms K-means for aviation incidents, but performs poorly for seminar announcements. GMM and K-means achieve similar results, which is not surprising because K-means can be viewed as a special case of the spherical GMM we used (Duda et al., 2001).

Overall speaking, results show that GMM is the best among the four generative models for the distri-

(a) Results on **seminar announcements**. No macro- and micro-average result is reported for NB and BMM as they merged the *etime* cluster with the *stime* cluster. Numbers in brackets are the respective measures of the *stime* cluster when evaluated for *etime*.

Model	*stime*	*etime*	*speaker*	*location*	Macro-avg	Micro-avg	Parameter
NB	0.558	(0.342)	0.276	0.172	—	—	Nil
BMM	0.822	(0.440)	0.412	0.402	—	—	Nil
GMM	0.450	0.530	0.417	0.426	0.557	0.455	Nil
LNLR	0.386	0.239	0.200	0.208	0.264	0.266	β = .0005
K-means	0.560	0.574	0.335	0.426	0.538	0.452	Nil

(b) Results on **aviation incidents**. Target slots are *airline (AL)* , *flight number (FN)*, *aircraft model (AC)*, *location (LO)* and *country (CO)*.

Model	*AL*	*FN*	*AC*	*LO*	*CO*	Macro-avg	Micro-avg	Parameter
NB	0.896	0.473	0.676	0.504	0.533	0.618	0.628	Nil
BMM	0.862	0.794	0.656	0.695	0.614	0.741	0.724	Nil
GMM	0.859	0.914	0.635	0.576	0.538	0.730	0.692	Nil
LNLR	0.597	0.352	0.314	0.286	0.291	0.379	0.396	β = .0005
K-means	0.859	0.936	0.661	0.576	0.538	0.729	0.701	Nil

(c) Results on **management succession events**. Target slots are *person joining (PersonIn)*, *person leaving (PersonOut)*, *organization (Org)*, and *position (Post)*.

Model	*PersonIn*	*PersonOut*	*Org*	*Post*	Macro-avg	Micro-avg	Parameter
NB	0.545	0.257	0.473	0.455	0.459	0.437	Nil
BMM	0.550	0.437	0.800	0.767	0.650	0.648	Nil
GMM	0.583	0.432	0.813	0.803	0.679	0.676	Nil
LNLR	0.419	0.245	0.319	0.399	0.351	0.346	β = .0002
K-means	0.372	0.565	0.835	0.814	0.645	0.665	Nil

Table 1: Performance summary of the different generative models and K-means in terms of F1.

Data set	Parameter	Value
Seminar announcements	$\{\lambda_k\}_{k=1}^4$	$\{2\}_{k=1}^4$
	$\{\tau_k\}_{k=1}^4$	$\{1\}_{k=1}^4$
Aviation incidents	$\{\lambda_k\}_{k=1}^5$	$\{1\}_{k=1}^5$
	$\{\tau_k\}_{k=1}^5$	$\{1\}_{k=1}^5$
Management succession	$\{\lambda_k\}_{k=1}^4$	{1,2,2,2}
	$\{\tau_k\}_{k=1}^4$	{1,2,2,2}

Table 2: Parameter settings for $p(\boldsymbol{y}_i; \Lambda)$.

bution $p(x|y; \Theta)$. We proceed with incorporating the proposed prior into GMM for further explorations.

Effectiveness of the proposed prior

We evaluate the effectiveness of the proposed prior by combining it with GMM. Specifically, the combined model follows Eq. (1), with $p(\boldsymbol{y}_i; \Lambda)$ computed using the Poisson-based formula in Eq. (3) and $p(x_{i,j}|y_{i,j}; \Theta)$ following Eq. (6) as in GMM.

We empirically determine the parameters used in $p(\boldsymbol{y}_i; \Lambda)$ to maximize data's log-likelihood as noted. Table 2 reports the values of $\{\lambda_k\}_{k=1}^{K-1}$ and $\{\tau_k\}_{k=1}^{K-1}$ for different data sets. Recall that λ_k specifies the *maximum* number of candidates that the k-th slot can generate, and its value is observed to be small in real data. τ_k specifies the *expected* number of candidates that the k-th slot will generate.

Table 3 reports the performance of the combined model ("GMM with prior") on the three data sets, along with results of GMM and K-means for easy comparison. The combined model improves over both GMM and K-means for seminar announcements and aviation incidents, as can be seen from the models' macro- and micro-average performance.

The advantages brought by the proposed prior are mainly reflected in slots that are difficult to cluster under GMM and K-means. Taking seminar announcements as an example, GMM and K-means achieve high precision but low recall for *stime*, and low precision but high recall for *etime*. When examining the clusters produced by these two models, we found one small cluster that contains mostly *stime* fillers (thus high precision but low recall), and another much larger cluster that contains mostly *etime* fillers together with most of the remaining *stime* fillers (thus low precision but high recall for *etime*).

(a) Results on **seminar announcements**.

Model	Metric	*stime*	*etime*	*speaker*	*location*	Macro-avg	Micro-avg
GMM with Prior	P	0.964	0.983	0.232	0.253	0.608	0.416
	R	0.680	0.932	0.952	0.481	0.761	0.738
	F1	0.798	0.957	0.374	0.331	0.676	0.532
GMM	P	1.000	0.362	0.300	0.436	0.524	0.407
	R	0.291	0.984	0.686	0.416	0.594	0.518
	F1	0.450	0.530	0.417	0.426	0.557	0.455
K-means	P	0.890	0.434	0.222	0.436	0.496	0.389
	R	0.408	0.847	0.679	0.416	0.588	0.541
	F1	0.560	0.574	0.335	0.426	0.538	0.452

(b) Results on **aviation incidents**.

Model	Metric	*AL*	*FN*	*AC*	*LO*	*CO*	Macro-avg	Micro-avg
GMM with Prior	P	1.000	1.000	1.000	0.741	0.833	0.915	0.908
	R	0.753	0.877	0.465	0.588	0.727	0.682	0.673
	F1	0.859	0.935	0.635	0.656	0.777	0.782	0.773
GMM	P	1.000	1.000	1.000	0.563	0.433	0.799	0.724
	R	0.753	0.842	0.465	0.588	0.709	0.672	0.664
	F1	0.859	0.914	0.635	0.576	0.538	0.730	0.692
K-means	P	1.000	0.981	0.830	0.563	0.433	0.761	0.711
	R	0.753	0.895	0.549	0.588	0.709	0.699	0.691
	F1	0.859	0.936	0.661	0.576	0.538	0.729	0.701

(c) Results on management succession events.

Model	Metric	*PersonIn*	*PersonOut*	*Org*	*Post*	Macro-avg	Micro-avg
GMM with Prior	P	0.458	0.610	0.720	0.774	0.640	0.642
	R	0.784	0.352	0.969	0.846	0.738	0.731
	F1	0.578	0.447	0.826	0.809	0.686	0.683
GMM	P	0.464	0.605	0.725	0.792	0.647	0.648
	R	0.782	0.336	0.925	0.815	0.715	0.707
	F1	0.583	0.432	0.813	0.803	0.679	0.676
K-means	P	0.382	0.515	0.733	0.839	0.607	0.639
	R	0.363	0.625	0.969	0.791	0.687	0.693
	F1	0.372	0.565	0.835	0.814	0.645	0.665

Table 3: Comparison between the combined model (GMM with the proposed prior), GMM and K-means.

This shows that GMM, when used with the multinomial prior, and K-means have difficulties separating candidates from these two slots. In contrast, the combined model improves the recall of *stime* to 68%, as compared to 29.1% achieved by GMM with the multinomial prior and 40.8% by K-means, without sacrificing precision. It also improves the precision of *etime* from 36.2% to 98.3%.

For aviation incidents, the advantage of the proposed prior is reflected in the *location (LO)* and *country (CO)* slots, which may confuse the various models as they both belong to the entity type *location*. The proposed prior improves the precision of these two slots greatly by trying to distribute them into appropriate slots in the clustering process.

The three models achieve very similar performance on management succession events as Table 3(c) shows. Surprisingly, incorporating the Poisson-based prior into GMM does not seem useful in separating *PersonIn* and *PersonOut* slot fillers. To investigate the possible reasons for this, we examine feature values in the centriods of the two clusters learned by the three models.

Tables 4 and 5 respectively list the top-10 features in the *PersonIn* cluster and the *PersonOut* cluster learned by the combined model[1], and their corresponding values in the centriods of the two clusters. The two clusters share 3 of the top-5 features, some

[1]We made similar observations from centriods learned in GMM and K-Means, which are therefore not reported here.

Top-10 features	Values in the centriod of: *PersonIn*	*PersonOut*
type:⟨person⟩	0.9985	1
unigram after:,	0.7251	0.3404
unigram before:⟨s⟩	0.2705	0
bigram after:, ⟨digits⟩	0.2105	0.1879
bigram after:, who	0.1404	0.0567
unigram before:,	0.1067	0.0035
dobj:succeeds	0.0906	0
unigarm before:succeeds	0.0892	0
nsubj:resigned	0.0746	0.0284
unigram before:said	0.0673	0

Table 4: Top-10 features in the *PersonIn* cluster, as learned by GMM with the proposed prior.

Top-10 features	Values in the centriod of: *PersonOut*	*PersonIn*
type:⟨person⟩	1	0.9985
unigram before:mr.	0.9894	0
bigram before:⟨s⟩ mr.	0.5213	0
unigram after:,	0.3404	0.7251
bigram after:, ⟨digits⟩	0.1879	0.2105
unigram after:was	0.1667	0.0556
nsubj:president	0.1667	0.0117
nsubj:succeeds	0.1028	0.0102
bigram before:, mr.	0.0957	0
unigram after:'s	0.0745	0.0073

Table 5: Top-10 features in the *PersonOut* cluster, as learned by GMM with the proposed prior.

of them being general context features that might not help characterizing candidates from different slots (e.g. the unigram after the candidate is a comma). Both lists also contain features from dependency parse trees. Note that the "dobj:succeeds" feature in the *PersonIn* cluster is in fact contributed by *PersonOut* slot fillers, while the "nsubj:succeeds" feature in the *PersonOut* cluster is contributed by *PersonIn* slot fillers. Although listed among the top-10, these features have relatively low values in the learned centriods (about 0.1). These observations may suggest that the management succession data set lacks strong, discriminative features for all models to effectively distinguish between *PersonIn* and *PersonOut* candidates in an unsupervised manner.

To conclude, the proposed prior is effective in assigning different but confusing candidate slot fillers into appropriate slots, when there exist reasonable features that can be exploited in the label assignment process. This is evident by the improvements the proposed prior brings to GMM in the seminar announcement and aviation incident data sets.

8 Conclusions

We propose a generative model that incorporates distributional prior knowledge about template slots in a document for the unsupervised IE task. Specifically, we propose a Poisson-based prior that prefers label assignments to cover more distinct slots in the same document. The proposed prior also allows a slot to generate multiple fillers in a document, up to a certain number of times depending on the domain of interest.

We experimented with four existing generative models for the task of clustering slot fillers with a multinomial prior, which assumes that labels are generated independently in a document. We then evaluate the effectiveness of the proposed prior by incorporating it into the Gaussian mixture model (GMM), which is shown to be the best among the four existing models in our experiments. By incorporating the proposed prior into GMM, we can obtain significantly better clustering results on two out of three data sets.

Further improvements to this work are possible. Firstly, we assume that some adjustable parameters in the proposed prior can be manually fixed, such as the number of template slots in the output and the maximum numbers of fillers that can be generated by different slots. We are looking into methods for automatically learning such parameters. This will help improve the applicability of our work to different domains as an unsupervised model. Secondly, we currently consider in the prior a probability distribution over all possible label assignments for every document. This can be computationally expensive if input documents are long, or when we aim to discover large templates with large values of K. An alternative is to consider an approximate solution that evaluates, for instance, only the top few label assignments that are likely to maximize the likelihood of our observations. This remains as an interesting future work of this study.

Acknowledgments

This work is supported by DSO National Laboratories. We thank the anonymous reviewers for their helpful comments.

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *International Joint Conference on Artificial Intelligence*, pages 2670–2676.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602.

Richard O. Duda, Peter E. Hart, and David G. Stork. 2001. *Pattern classification*. Wiley-Interscience, 2nd edition.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 207–214, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370.

Dayne Freitag and Andrew Kachites McCallum. 1999. Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*.

João Graça, Kuzman Ganchev, and Ben Taskar. 2007. Expectation maximization and posterior constraints. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 415, Morristown, NJ, USA. Association for Computational Linguistics.

J. B. Macqueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1*, pages 281–297.

Zvika Marx, Ido Dagan, and Eli Shamir. 2002. Cross-component clustering for template induction. In *Proceedings of the 2002 ICML Workshop on Text Learning*.

MUC-6. 1995. *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufmann, San Francisco, CA.

Benjamin Rosenfeld and Ronen Feldman. 2006. URES : An unsupervised Web relation extraction system. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 667–674.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of the COLING/ACL Main conference poster sessions*, pages 731–738.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 304–311.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 224–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wikipedia. 2009. List of accidents and incidents involving commercial aircraft. http://en.wikipedia.org/wiki/List_of_accidents_and_incidents_involving_commercial_aircraft.

Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining Wikipedia texts using information from the web. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*.

Relation Acquisition using Word Classes and Partial Patterns

Stijn De Saeger[†*] **Kentaro Torisawa**[†] **Masaaki Tsuchida**[§] **Jun'ichi Kazama**[†]
Chikara Hashimoto[†] **Ichiro Yamada**[‡] **Jong Hoon Oh**[†] **István Varga**[†] **Yulan Yan**[†]

[†] Information Analysis Laboratory, National Institute of
Information and Communications Technology, 619-0289 Kyoto, Japan
{stijn,torisawa,kazama,ch,rovellia,istvan,yulan}@nict.go.jp

[§] Information and Media Processing Laboratories, NEC Corporation, 630-0101 Nara, Japan
m-tsuchida@cq.jp.nec.com

[‡] Human & Information Science Research Division,
NHK Science & Technology Research Laboratories, 157-8510 Tokyo, Japan
yamada.i-hy@nhk.or.jp

Abstract

This paper proposes a semi-supervised relation acquisition method that does not rely on extraction patterns (e.g. "*X causes Y*" for causal relations) but instead learns a combination of indirect evidence for the target relation — *semantic word classes* and *partial patterns*. This method can extract long tail instances of semantic relations like causality from rare and complex expressions in a large Japanese Web corpus — in extreme cases, patterns that occur only once in the entire corpus. Such patterns are beyond the reach of current pattern based methods. We show that our method performs on par with state-of-the-art pattern based methods, and maintains a reasonable level of accuracy even for instances acquired from infrequent patterns. This ability to acquire long tail instances is crucial for risk management and innovation, where an exhaustive database of high-level semantic relations like causation is of vital importance.

1 Introduction

Pattern based relation acquisition methods rely on *lexico-syntactic patterns* (Hearst, 1992) for extracting relation instances. These are templates of natural language expressions such as "X causes Y" that signal an instance of some semantic relation (i.e., causality). Pattern based methods (Agichtein and Gravano, 2000; Pantel and Pennacchiotti, 2006b; Paşca et al., 2006; De Saeger et al., 2009) learn many such patterns to extract new instances (word pairs) from the corpus.

However, since extraction patterns are learned using statistical methods that require a certain frequency of observations, pattern based methods fail to capture relations from complex expressions in which the pattern connecting the two words is rarely observed. Consider the following sentence:

> "Curing hypertension alleviates the deterioration speed of the renal function, thereby lowering the risk of causing intracranial bleeding"

Humans can infer that this sentence expresses a causal relation between the underlined noun phrases. But the actual *pattern* connecting them, i.e., "*Curing X alleviates the deterioration speed of the renal function, thereby lowering the risk of causing Y*", is rarely observed more than once even in a 10^8 page Web corpus. In the sense that the term *pattern* implies a recurring event, this expression contains no pattern for detecting the causal relation between *hypertension* and *intracranial bleeding*. This is what we mean by "long tail instances" — words that co-occur infrequently, and only in sparse extraction contexts.

Yet an important application of relation extraction is mining the Web for so-called *unknown unknowns* — in the words of D. Rumsfeld, "things we don't know we don't know" (Torisawa et al., 2010). In knowledge discovery applications like risk management and innovation, the usefulness of relation extraction lies in its ability to find many unexpected remedies for diseases, causes of social problems, and so on. To give an example, our relation extrac-

[*] This work was done when all authors were at the National Institute of Information and Communications Technology.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 825–835,
Edinburgh, Scotland, UK, July 27–31, 2011.

tion system found a blog post mentioning Japanese automaker Toyota as a hidden cause of Japan's deflation. Several months later the same connection was made in an article published in an authoritative economic magazine.

We propose a semi-supervised relation extraction method that does not rely on direct pattern evidence connecting the two words in a sentence. We argue that the role of binary patterns can be replaced by a combination of two types of *indirect* evidence: *semantic class information* about the target relation and *partial patterns*, which are *fragments* or *subpatterns* of binary patterns. The intuition is this: if a sentence like the example sentence above contains some word X belonging to the class of *medical conditions* and another word Y from the class of *traumas*, and X matches the partial pattern "*... causing X*", there is a decent chance that this sentence expresses a causal relation between X and Y. We show that just using this combination of indirect evidence we can pick up semantic relations with roughly 50% precision, regardless of the complexity or frequency of the expression in which the words co-occur. Furthermore, by combining this idea with a straightforward machine learning approach, the overall performance of our method is on par with state-of-the-art pattern based methods. However, our method manages to extract a large number of instances from sentences that contain no pattern that can be learned by pattern induction methods.

Our method is a two-stage system. Figure 1 presents an overview. In Stage 1 we apply a state-of-the-art pattern based relation extractor to a Web corpus to obtain an initial batch of relation instances. In Stage 2 a supervised classifier is built from various components obtained from the output of Stage 1. Given the output of Stage 1 and access to a Web corpus, the Stage 2 extractor is completely self-sufficient, and the whole method requires no supervision other than a handful of seed patterns to start the first stage extractor. The whole procedure is therefore minimally supervised. Semantic word classes and partial patterns play a crucial role throughout all steps of the process.

We evaluate our method on three relation acquisition tasks (*causation*, *prevention* and *material* relations) using a 600 million Japanese Web page corpus (Shinzato et al., 2008) and show that our system can successfully acquire relations from both frequent and infrequent patterns. Our system extracted 100,000 causal relations with 84.6% precision, 50,000 prevention relations with 58.4% precision and 25,000 material relations with 76.1% precision. In the extreme case, we acquired several thousand word pairs co-occurring only in patterns that appear once in the entire corpus. We call such patterns *single occurrence* (SO) patterns. Word pairs that co-occur only with SO patterns represent the theoretical limiting case of relations that cannot be acquired using existing pattern based methods. In this sense our method can be seen as complementary with pattern based approaches, and merging our method's output with that of a pattern based method may be beneficial.

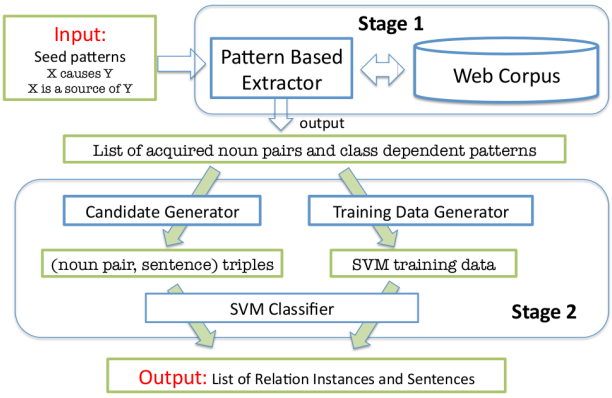


Figure 1: Proposed method: data flow.

2 Stage 1 Extractor

This section introduces our Stage 1 extractor: the pattern based method from (De Saeger et al., 2009), which we call CDP for "*class dependent patterns*". We give a brief overview below, and refer the reader to the original paper for a more comprehensive explanation.

CDP takes a set of *seed patterns* as input, and automatically learns new *class dependent patterns* as paraphrases of the seed patterns. Class dependent patterns are semantic class restricted versions of ordinary lexico-syntactic patterns. Existing methods use class *independent* patterns such as "X causes Y" to learn causal relations between X and Y. Class dependent patterns however place semantic class re-

strictions on the noun pairs they may extract, like "$Y_{accidents}$ *causes* $X_{incidents}$". The *accidents* and *incidents* subscripts specify the semantic class of the X and Y slot fillers.

These class restrictions make it possible to distinguish between multiple senses of highly ambiguous patterns (so-called "generic" patterns). For instance, given the generic pattern "Y *by* X", if we restrict Y and X in to the semantic classes of *injuries* and *accidents* (as in "*death by drowning*"), the class dependent pattern "$Y_{injuries}$ *by* $X_{accidents}$" becomes a valid paraphrase of "X *causes* Y" and can safely be used to extract causal relations, whereas other class dependent versions of the same generic pattern (e.g., "$Y_{products}$ *by* $X_{companies}$", as in "*iPhone by Apple*") may not.

CDP ranks each noun pair in the corpus according to a score that reflects its likelihood of being a proper instance of the target relation, by calculating the semantic similarity of a set of seed patterns to the class dependent patterns this noun pair co-occurs with. The output of CDP is a list of noun pairs ranked by score, together with the highest scoring class dependent pattern each noun pair co-occurs with. This list becomes the input to Stage 2 of our method, as shown in Figure 1. We adopted CDP as Stage 1 extractor because, besides having generally good performance, the class dependent patterns provide the two fundamental ingredients for Stage 2 of our method — the target semantic word classes for a given relation (in the form of the semantic class restrictions attached to patterns), and partial patterns.

To obtain fine-grained semantic word classes we used the large scale word clustering algorithm from (Kazama and Torisawa, 2008), which uses the EM algorithm to compute the probability that a word w belongs to class c, i.e., $P(c|w)$. Probabilistic clustering defines no discrete boundary between members and non-members of a semantic class, so we simply assume w belongs to c whenever $P(c|w) \geq 0.2$. For this work we clustered 10^6 nouns into 500 classes.

Finally, we adopt the structural representation of patterns introduced in (Lin and Pantel, 2001). All sentences in our corpus are dependency parsed, and patterns consist of words on the path of dependency relations connecting two nouns.

3 Stage 2 Extractor

We use CDP as our Stage 1 extractor, and the top N noun pairs along with the class dependent patterns that extract them are given as input to Stage 2, which represents the main contribution of this work. As shown in Figure 1, Stage 2 consists of three modules: a *candidate generator*, a *training data generator* and a *supervised classifier*. The training data generator builds training data for the classifier from the top N output of CDP and sentences retrieved from the Web corpus. This classifier then scores and ranks the candidate relations generated by the candidate relation generator. We introduce each module below.

Candidate Generator This module generates sentences containing candidate word pairs for the target relation from the corpus. It does so using the semantic class restrictions and partial patterns obtained from the output of CDP. The set of all semantic class pairs obtained from the class dependent patterns that extracted the top N results become the target semantic class pairs from which new candidate instances are generated. We extract all sentences containing a word pair belonging to one of the target class pairs from the corpus.

From these sentences we keep only those that contain a trace of evidence for the target semantic relation. For this we decompose the class dependent patterns from the Stage 1 extractor into *partial patterns*. As mentioned previously, patterns consist of words on the path of dependency relations connecting the two target words in a syntactic tree. To obtain partial patterns we split this dependency path into its two constituent branches, each one leading from the leaf word (i.e. variable) to the syntactic head of the pattern. For example, "$X \xleftarrow{\text{subj}}$ *causes* $\xrightarrow{\text{obj}} Y$" is split into two partial patterns "$X \xleftarrow{\text{subj}}$ *causes*" and "*causes* $\xrightarrow{\text{obj}} Y$". These partial patterns capture the predicate structures in binary patterns.[1] We discard partial patterns with syntactic heads other than verbs or adjectives.

The candidate genarator retrieves all sentences from the corpus in which two nouns belonging to one of the target semantic classes co-occur and

[1] In Japanese, case information is encoded in post-positions attached to the noun.

where at least one of the nouns matches a partial pattern. As shown in Figure 1, these sentences and the candidate noun pairs they contain (called *(noun pair, sentence)* triples hereafter) are submitted to the classifier for scoring. Restricting candidate noun pairs by this combination of semantic word classes and partial pattern matching proved to be quite powerful. For instance, in the case of causal relations we found that close to 60% of the *(noun pair, sentence)* triples produced by the candidate generator were correct (Figure 6).

Training Data Generator As shown in Figure 1, the *(noun pair, sentence)* triples used as training data for the SVM classifier were generated from the top results of the Stage 1 extractor and the corpus. We consider the noun pairs in the top N output of the Stage 1 extractor as *true* instances of the target relation (even though they may contain erroneous extractions), and retrieve from the corpus all sentences in which these noun pairs co-occur and that match one of the partial patterns mentioned above. In our experiments we set N to $25,000$. We randomly select positive training samples from this set of *(noun pair, sentence)* triples.

Negative training samples are also selected randomly, as follows. If one member of the target noun pair in the positive samples above matches a partial pattern but the other does not, we randomly replace the latter by another noun found in the same sentence, and generate this new *(noun pair, sentence)* triple as a negative training sample. In the causal relation experiments this approach had about 5% chance of generating *false* negatives — noun pairs contained in the top N results of the Stage 1 extractor. Such samples were discarded. Our experimental results show that this scheme works quite well in practice. We randomly sample M positive and negative samples from the autogenerated training data to train the SVM. M was empirically set to 50,000 in our experiments.

SVM Classifier We used a straightforward feature set for training the SVM classifier. Because our classifier will be faced with sentences containing long and infrequent patterns where the distance between the two target nouns may be quite large, we did not try to represent lexico-syntactic patterns as features but deliberately restricted the feature set to local context features of the candidate noun pair in the target sentence. Concretely, we looked at bigrams and unigrams surrounding both nouns of the candidate relation, as the local context around the target words may contain many telling expressions like "*increase in X*" or "*X deficiency*" which are useful clues for causal relations. Also, in Japanese case information is encoded in post-positions attached to the noun, which is captured by the unigram features.

In addition to these base features, we include the semantic classes to which the candidate noun pair belongs, the partial patterns they match in this sentence, and the infix words inbetween the target noun pair. Note that this feature set is not intended to be optimal beyond the actual claims of this paper, and we have deliberately avoided exhaustive feature engineering so as not to obscure the contribution of semantic classes and partial pattern to our approach. Clearly an *optimal* classifier will incorporate many more advanced features (see GuoDong et al. (2005) for a comprehensive overview), but even without sophisticated feature engineering our classifier achieved sufficient performance levels to support our claims. An overview of the feature set is given in Table 1. The relative contribution of each type of features is discussed in section 4. In preliminary experiments we found a polynomial kernel of degree 3 gave the best results, which suggests the effectiveness of combining different types of indirect evidence.

The SVM classifier outputs *(noun pair, sentence)* triples, ranked by SVM score. To obtain the final output of our method we assign each unique noun pair the maximum score from all *(noun pair, sentence)* triples it occurs in, and discard all other sentences for this noun pair. In section 4 below we evaluate the acquired noun pairs in the context of the sentence that maximizes their score.

4 Evaluation

We demonstrate the effectiveness of semantic word classes and partial pattern matching for relation extraction by showing that the method proposed in this paper performs at the level of other state-of-the-art relation acquisition methods. In addition we demonstrate that our method can successfully extract relation instances from infrequent patterns, and we

Feature type	Description	Number of features
Morpheme features	Unigram and bigram morphemes surrounding both target words.	554,395
POS features	Coarse- and fine-grained POS tags of the noun pair and morpheme features.	2,411
Semantic features	Semantic word classes of the target noun pair.	1000 (500 classes $\times 2$)
Infix word features	Morphemes found inbetween the target noun pair.	94,448
Partial patterns	Partial patterns matching the target noun pair.	86

Table 1: Feature set used in the Stage 2 classifier, and their number for the causal relation experiments.

explore several criteria for what constitutes an infrequent pattern — including the theoretical limiting case of patterns observed only once in the entire corpus. These instances are impossible to acquire by pattern based methods. The ability to acquire relations from extremely infrequent expressions with decent accuracy demonstrates the utility of combining semantic word classes with partial pattern matching.

4.1 Experimental Setting

We evaluate our method on three semantic relation acquisition tasks: *causality*, *prevention* and *material*. Two concepts stand in a *causal relation* when the source concept (the "cause") is directly or indirectly responsible for the subsequent occurrence of the target concept (its "effect"). In a *prevention* relation the source concept directly or indirectly acts to avoid the occurrence of the target concept, and in a *material* relation the source concept is a material or ingredient of the target concept.

For our experiments we used the latest version of the TSUBAKI corpus (Shinzato et al., 2008), a collection of 600 million Japanese Web pages dependency parsed by the Japanese dependency parser KNP[2]. In our implementation of CDP, lexico-syntactic patterns consist of words on the path connecting two nouns in a dependency parse tree. We discard patterns from dependency paths longer than 8 constituent nodes. Furthermore, we estimated pattern frequencies in a subset of the corpus (50 million pages, or 1/12th of the entire corpus) and discarded patterns that co-occur with less than 10 unique noun pairs in this smaller corpus. These restrictions do not apply to the proposed method, which can extract noun pairs connected by patterns of arbitrary length, even if found only once in the corpus. For our purpose we treat dependency paths whose observed frequency is below this threshold as insufficiently frequent to be considered as "patterns". This threshold is of course arbitrary, but in section 4 we show that our results are not affected by these implementation details.

We asked three human judges to evaluate random *(noun pair, sentence)* triples, i.e. candidate noun pairs in the context of some corpus sentence in which they co-occur. If the judges find the sentence contains sufficient evidence that the target relation holds between the candidate nouns, they mark the noun pair correct. To evaluate the performance of each method we use two evaluation criteria: *strict* (all judges must agree the candidate relation is correct) and *lenient* (decided by the judges' majority vote). Over all experiments the interrater agreement (Kappa) ranged between 0.57 and 0.82 with an average of 0.72, indicating substantial agreement (Landis and Koch, 1977).

4.1.1 Methods Compared

We compare our results to two pattern based methods: CDP (the Stage 1 extractor) and *Espresso* (Pantel and Pennacchiotti, 2006a).

Espresso is a popular bootstrapping based method that uses a set of seed instances to induce extraction patterns for the target relation and then acquire new instances in an iterative bootstrapping process. In each iteration Espresso performs pattern induction, pattern ranking and selection using previously acquired instances, and uses the newly acquired patterns to extraction new instances. Espresso computes a reliability score for both instances and patterns based on their pointwise mutual information (PMI) with the top-scoring patterns and instances from the previous iteration.[3] We refer to (Pantel and

[2] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp.html

[3] In our implementation of Espresso we found that, despite the many parameters for controlling the bootstrapping process,

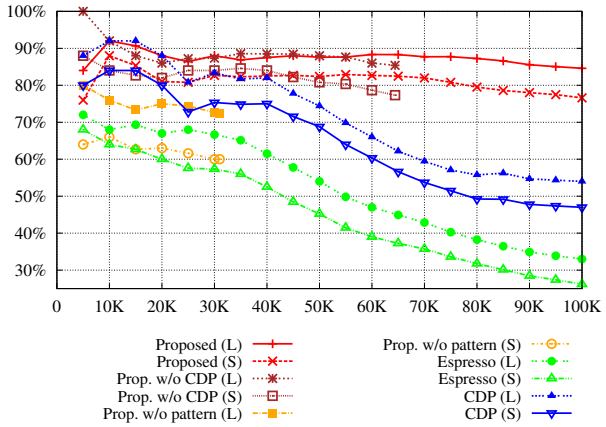


Figure 2: Precision of acquired relations (causality). L and S denote lenient and strict evaluation.

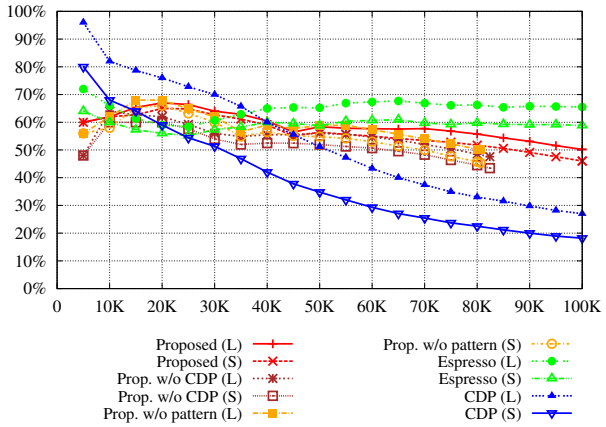


Figure 3: Precision of acquired relations (prevention). L and S denote lenient and strict evaluation.

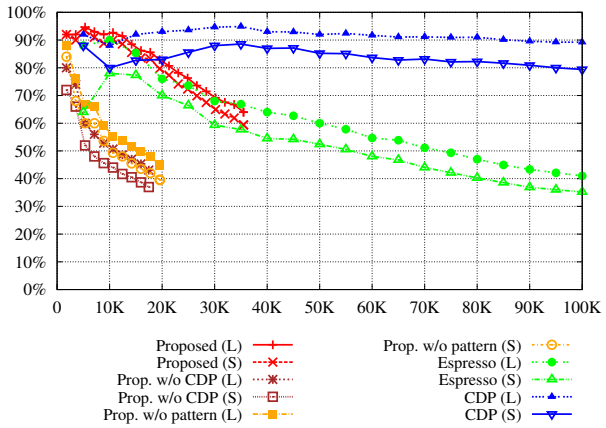


Figure 4: Precision of acquired relations (material). L and S denote lenient and strict evaluation.

Pennacchiotti, 2006a) for a more detailed description.

For all methods compared we rank the acquired noun pairs by their score and evaluated 500 random samples from the top 100,000 results. For noun pairs acquired by CDP and Espresso we select the pattern that extracted this noun pair (in the case of Espresso, the pattern with the highest PMI for this noun pair), and randomly select a sentence in which the noun pair co-occurs with that pattern from our corpus. For the proposed method we evaluate noun pairs in the context of the *(noun pair, sentence)* triple with the highest SVM score.

4.2 Results and Discussion

The performance of each method on the causality, prevention and material relations are shown in Figures 2, 3 and 4 respectively. In the causality experiments (Figure 2) the proposed method performs on par with CDP for the top 25,000 results, both achieving close to 90% precision. But whereas CDP's performance drops from there our method maintains the same high precision throughout (84.6%, lenient). Both our method and CDP outperform Espresso by a large margin.

For the prevention relation (Figure 3), precision is considerably lower for all methods except the top 10,000 of CDP (82% precision, lenient). The proposed method peaks at around 20,000 results (67% precision, lenient) and performance remains more or less constant from there on. The proposed method overtakes CDP's performance around the top 45,000 mark, which suggests that combining the results of both methods may be beneficial.

In the material relations the proposed method slightly outperforms both pattern based methods in the top 10,000 results (92% precision, lenient).

it remains very difficult to prevent *semantic drift* (Komachi et al., 2008) from occurring. One small adjustment to the algorithm stabilized the bootstrapping process considerably and gave overall better results. In the pattern induction step (section 3.2 in (Pantel and Pennacchiotti, 2006a)), Espresso computes a reliability score for each candidate pattern based on the weighted PMI of the pattern with all instances extracted so far. As the number of extracted instances increases this disproportionally favours high frequency (i.e. generic) patterns, so instead of using *all* instances for computing pattern reliability we only use the m most reliable instances from the previous iteration, which were used to extract the candidate patterns of the current iteration ($m = 200$, like the original).

However for this relation our method produced only 35,409 instances. The reason is that the top 25,000 results of CDP were all extracted by just 12 patterns, and these contained many patterns whose syntactic head is not a verb or adjective (like "*Y rich in X*" or "*Y containing X*"). Only 12 partial patterns were obtained, which greatly reduced the output of the proposed method. Taking into account the high performance of CDP for material relations, this suggests that for some relations our method's N and M parameters could use some tuning. In conclusion, in all three relations our method performs at a level comparable to state-of-the-art pattern based methods, which is remarkable given that it only uses indirect evidence.

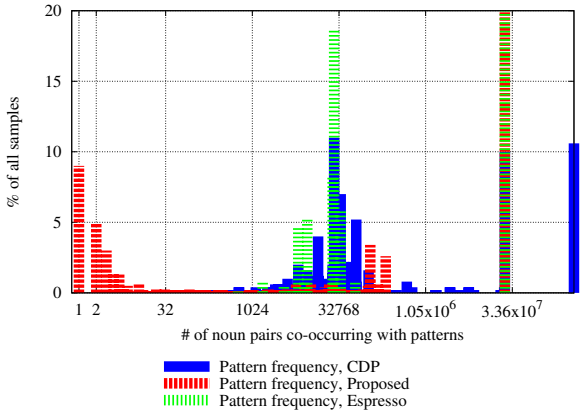


Figure 5: Frequencies of patterns in the evaluation data (causation).

Dealing with Difficult Extractions How does our method handle noun pairs that are difficult to acquire by pattern based methods? The graphs marked "Prop. w/o CDP" (Proposed without CDP) in Figures 2 , 3 and 4 show the number and precision of evaluated samples from the proposed method that do not co-occur in our corpus with any of the patterns that extracted the top N results of the first stage extractor. These graphs show that our method is not simply regenerating CDP's top results but actually extracts many noun pairs that do not co-occur in patterns that are easily learned. Figure 2 shows that roughly two thirds of the evaluated samples are in this category, and that their performance is not significantly worse than the overall result. The same conclusion holds for the prevention results (Figure 3), where over 80% of the proposed method's samples are noun pairs that do not co-occur with easily learnable patterns. Their precision is about 5% worse than all samples from the proposed method. For material relations (Figure 4) about half of all evaluated samples are in this category, but their precision is markedly worse compared to all results.

For genuinely *infrequent* patterns, the graphs marked "Prop. w/o pattern" (Proposed without pattern) in Figures 2 , 3 and 4 show the number and precision of noun pairs evaluated for the proposed method that were acquired from sentences without any discernible pattern. As explained in section 4 above, these constitute noun pairs co-occurring in a sentence in which the path of dependency relations connecting them is either longer than 8 nodes or can extract fewer than 10 noun pairs in 50 million Web pages. Note that in theory it is possible that these noun pairs could not be acquired by pattern based methods due to this threshold — patterns must be able to extract more than 10 different noun pairs in a subset of our corpus, while the proposed method does not have this constraint. So at least in theory, pattern based methods might be able to acquire all noun pairs obtained by our method by lowering this threshold. To see that this is unlikely to be the case, consider Figure 5, which shows the pattern frequency of the patterns induced by CDP and Espresso for the causality experiment. The x-axis represents pattern frequency in terms of the number of unique noun pairs a pattern co-occurs with in our corpus (on a log scale), and the y-axis shows the percentage of samples that was extracted by patterns of a given frequency.[4] Figure 5 shows that for the pattern based methods, the large majority of noun pairs was extracted by patterns that co-occur with several thousand different noun pairs. Extrapolating the original frequency threshold of 10 nounpairs to the size of our entire corpus roughly corresponds to about 120 distinct noun pairs (10 times in 1/12th of the entire corpus). In Figure 5, the histograms for the pattern based methods CDP and Espresso start around 1000 noun pairs, which is far above this new lowerbound.

[4] In the case of CDP we ignore semantic class restrictions on the patterns when comparing frequencies. For Espresso, the most frequent pattern ("Y by X" at the 24,889,329 data point on the x-axis) extracted up to 53.8% of the results, but the graph was cut at 20% for readability.

Causality	〈カテコラミン〉が心拍数の急上昇をもたらすことから、血管内の血行状態の変化が血管内障害につながり、[血栓形成] を促進する。 Because ⟨catecholamine⟩ causes a rapid increase of heart rate, the change of circulation inside the blood vessels leads to blood vessel disorders and promotes [thrombus generation].
	〈頻脈発作〉が始まってキシロカインを静注したところ、患者さんが突然意識をなくして [全身痙攣] を起こしたということです。 When we injected Xylocaine during a ⟨tachycardia seizure⟩, the patient suddenly lost consciousness and fell into a fit of [convulsions].
	その理由としては〈動物性たんぱく質〉を多く取ることで [わきが] の原因物質が増加するからです。 (. . .) The reason is that by taking a lot of ⟨animal proteins⟩ the causative agents of [tragomaschalia] increase.
	* 抗酸化機能を高め、老化や〈生活習慣病〉の原因となる活性化酸素を消去する作用がある [ラドン]。 * [Radon] heightens the (body's) antioxidative function and is effective for eliminating activated oxygen, which is a cause of aging and ⟨lifestyle-related⟩ diseases.
Prevention	マグロのトロ、中トロの部分には、ＤＨＡや〈ＥＰＡ〉が豊富に含まれているので、[神経痛] の予防食材として有効です。 Because the fatty meat of tuna contains DHA and ⟨EPA⟩ in abundance, it is effective for preventing [neuralgia].
	空気の代わりに〈窒素ガス〉を使用すれば [粉塵爆発] の予防になります。 If you use ⟨nitrogen gas⟩ instead of air you may prevent [dust explosions].
	ヨーロッパでは古くから〈クミスクチン〉のお茶は「やせるお茶である」といわれ中性脂肪や [成人病] の予防に良いそうです。 In ancient Europe ⟨orthosiphon aristatus⟩ tea was called a "diet tea", and supposedly it helps preventing triglycerides and [adult diseases].
	*〈幌〉の収納時にスクリーンの間に挟んで、スクリーンの [スリキズ] を防ぐものです。 * (It) is something that prevents [scratches] on the screen if the ⟨calash⟩ gets stuck between the screens during storage.

Table 2: Causality and Prevention relations acquired from Single Occurrence (SO) patterns. ⟨X⟩ and [Y] indicate the relation instance's source and target words, and "*" indicates erroneous extractions.

Thus, pattern based methods naturally tend to induce patterns that are much more frequent than the range of patterns our method can capture, and it is unlikely that this is a result of implementation details like pattern frequency threshold.

The precision of noun pairs in the category "Prop. w/o pattern" is clearly lower than the overall results, but the graphs demonstrate that our method still handles these difficult cases reasonably well. The 500 samples evaluated contained 155 such instances for causality, 403 for prevention and 276 for material. For prevention, the high ratio of these noun pairs helps explain why the overall performance was lower than for the other relations.

Finally, the theoretical limiting case for pattern based algorithms consists of patterns that only co-occur with a single noun pair in the entire corpus (*single occurrence* or SO patterns). Pattern based methods learn new patterns that share many noun pairs with a set of reliable patterns in order to extract new relation instances. If a noun pair that co-occurs with a SO pattern also co-occurs with more reliable patterns there is no need to learn the SO pattern. If that same noun pair does *not* co-occur with any other reliable pattern, the SO pattern is beyond the reach of *any* pattern induction method. Thus, SO patterns are effectively useless for pattern based methods. For the 500 samples evaluated from the causality and prevention relations acquired by our method we found 7 causal noun pairs that co-occur only in SO patterns and 29 such noun pairs for prevention. The precision of these instances was 42.9% and 51.7% respectively. In total we found 8,716 causal noun pairs and 7,369 prevention noun pairs that co-occur only with SO patterns. Table 2 shows some example relations from our causality and prevention experiments that were extracted from SO patterns. To conclude, our method is able to acquire correct relations even from the most extreme infrequent expressions.

Semantic Classes, Partial Patterns or Both? In the remainder of this section we look at how the combination of semantic word classes and partial patterns benefits our method. For each relation we evaluated 1000 random *(noun pair, sentence)* triples satisfying the two conditions from section 3 — matching semantic class pairs and partial patterns. Surprisingly, the precision of these samples was 59% for causality, 40% for prevention and 50.4% for material, showing just how compelling these two types of indirect evidence are in combination.

To estimate the relative contribution of each heuristic we compared our candidate generation method against two baselines. The first baseline evaluates the precision of random noun pairs from

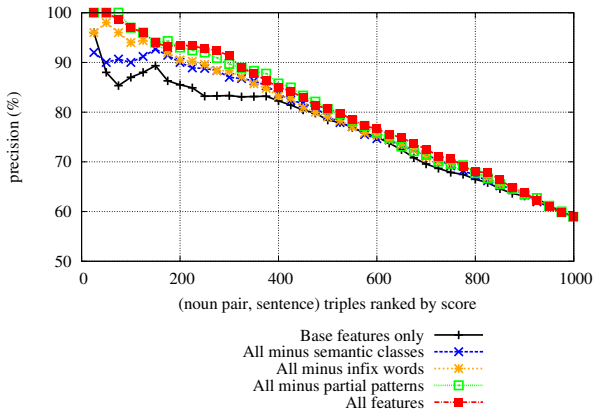


Figure 6: Contribution of feature sets (causality).

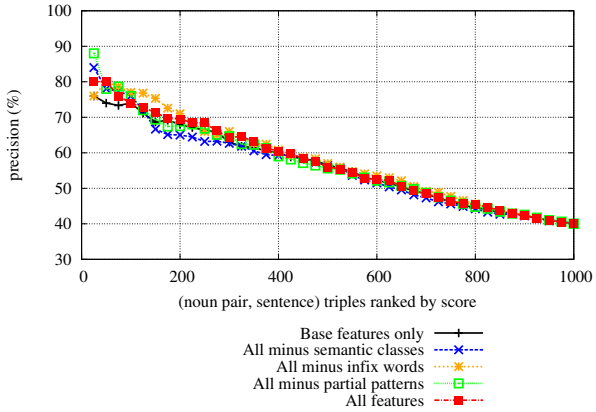


Figure 7: Contribution of feature sets (prevention).

the target semantic classes that co-occur in a sentence. The second baseline does the same for the second heuristic, selecting random sentences containing a noun pair that matches some partial pattern. Evaluating 100 samples for causality and prevention, we found the precision of the semantic class baseline was 16% for causality and 5% for prevention. The pattern fragment baseline gave 9% for causality and 22% for prevention. This is considerably lower than the precision of random samples that satisfy both the semantic class and partial pattern conditions, showing that the combination of semantic classes and partial patterns is more effective than either one individually.

Finally, we investigated the effect of the various feature sets used in the classifier. Figures 6, 7 and 8 show the results for the respective semantic relations. The "Base features" graph shows the performance the unigram, bigram and part-of-speech features. "All features" uses all features in Table 1. The other graphs show the effect of removing one type of features. These graphs suggest that the contribution of the individual feature types (semantic class information, partial patterns or infix words) to the classification performance is relatively minor, but in combination they do give a marked improvement over the base features, at least for some relations like causation and material. In other words, the main contribution of semantic word classes and partial patterns to our method's performance lies not in the final classification step but seems to occur at earlier stages of the process, in the candidate and training data generation steps.

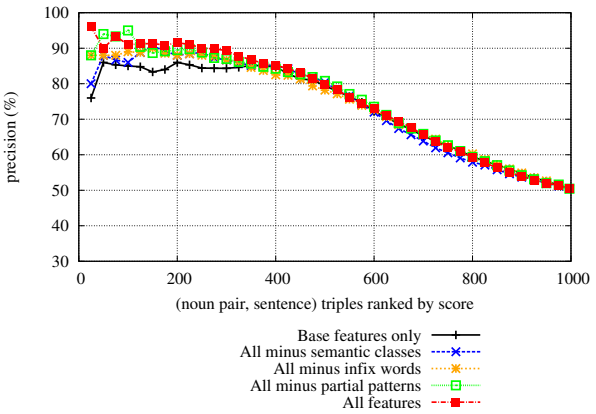


Figure 8: Contribution of feature sets (material).

5 Related Work

Using lexico-syntactic patterns to extract semantic relations was first explored by Hearst (Hearst, 1992), and has inspired a large body of work on semi-supervised relation acquisition methods (Berland and Charniak, 1999; Agichtein and Gravano, 2000; Etzioni et al., 2004; Pantel and Pennacchiotti, 2006b; Paşca et al., 2006; De Saeger et al., 2009), two of which were used in this work.

Some researchers have addressed the sparseness problems inherent in pattern based methods. Downey et al. (2007) starts from the output of the unsupervised information extraction system TextRunner (Banko and Etzioni, 2008), and uses language modeling techniques to estimate the reliability of sparse extractions. Paşca et al. (2006) alle-

viates pattern sparseness by using infix patterns that are generalized using classes of distributionally similar words. In addition, their method employs clustering based semantic similarities to filter newly extracted instances in each iteration of the bootstrapping process. A comparison with our method would have been instructive, but we were unable to implement their method because the original paper contains insufficient detail to allow replication.

There is a large body of research in the *supervised* tradition that does not use explicit pattern representations — kernel based methods (Zelenko et al., 2003; Culotta, 2004; Bunescu and Mooney, 2005) and CRF based methods (Culotta et al., 2006). These approaches are all fully supervised, whereas in our work the automatic generation of candidates and training data is an integral part of the method. An interesting alternative is distant supervision (Mintz et al., 2009), which trains a classifier using an existing database (Freebase) containing thousands of semantic relations, with millions of instances. We believe our method is more general, as depending on external resources like a database of semantic relations limits both the range of semantic relations (i.e., Freebase contains only relations between named entities, and none of the relations in this work) and languages (i.e., no resource comparable to Freebase exists for Japanese) to which the technology can be applied. Furthermore, it is unclear whether distant supervision can deal with noisy input such as automatically acquired relation instances.

Finally, inference based methods (Carlson et al., 2010; Schoenmackers et al., 2010; Tsuchida et al., 2010) are another attempt at relation acquisition that goes beyond pattern matching. Carlson et al. (2010) proposed a method based on inductive logic programming (Quinlan, 1990). Schoenmackers et al. (2010) takes relation instances produced by TextRunner (Banko and Etzioni, 2008) as input and induces first-order Horn clauses, and new instances are infered using a Markov Logic Network (Richardson and Domingo, 2006; Huynh and Mooney, 2008). Tsuchida et al. (2010) generated new relation hypotheses by substituting words in seed instances with distributionally similar words. The difference between these works and ours lies in the treatment of evidence. While the above methods learn inference rules to acquire new relation instances from independent information sources scattered across different Web pages, our method takes the other option of working with all the clues and indirect evidence a single sentence can provide. In the future, a combination of both approaches may prove beneficial.

6 Conclusion

We have proposed a relation acquisition method that is able to acquire semantic relations from infrequent expressions by focusing on the evidence provided by semantic word classes and partial pattern matching instead of direct extraction patterns. We experimentally demonstrated the effectiveness of this approach on three relation acquisition tasks, causality, prevention and material relations. In addition we showed our method could acquire a significant number of relation instances that are found in extremely infrequent expressions, the most extreme case of which are *single occurrence patterns*, which are beyond the reach of existing pattern based methods. We believe this ability is of crucial importance for acquiring valuable long tail instances. In future work we will investigate whether the current framework can be extended to acquire inter-sentential relations.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *Proc. of the fifth ACM conference on Digital libraries*, pages 85–94.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proc. of the 46th ACL-08:HLT*, pages 28–36.

Matthew Berland and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 57–64, College Park, Maryland, USA, June.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '05)*, pages 724–731.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for neverending language learning. In *Proc of the 24th AAAI*, pages 1306–1313.

Aron Culotta, Andrew McCallum, and Jonathan Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL)*, pages 296–303.

Aron Culotta. 2004. Dependency tree kernels for relation extraction. In *In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04*, pages 423–429.

Stijn De Saeger, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large Scale Relation Acquisition Using Class Dependent Patterns. In *Proc. of the 9th International Conference on Data Mining (ICDM)*, pages 764–769.

Doug Downey, Stefan Schoenmackers, and Oren Etzioni. 2007. Sparse information extraction: Unsupervised language models to the rescue. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL2007)*.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel Weld, and Alexander Yates. 2004. Web-scale information extraction in KnowItAll. In *Proc. of the 13th international conference on World Wide Web (WWW04)*, pages 100–110.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proc. of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 419–444.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th International Conference on Computational Linguistics (COLING'92)*, pages 539–545.

Tuyen N. Huynh and Raymond J. Mooney. 2008. Discriminative structure and parameter learning for markov logic networks. In *Proc. of the 25th ICML*, pages 416–423.

Jun'ichi Kazama and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 407–415.

Mamoru Komachi, Taku Kudo, Masashi Shimbo, and Yuji Matsumoto. 2008. Graph-based analysis of semantic drift in espresso-like bootstrapping algorithms. In *Proc. of EMNLP'08. Honolulu, USA*, pages 1011–1020.

Dekang Lin and Patrick Pantel. 2001. Dirt - discovery of inference rules from text. In *Proc. of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Names and Similarities on the Web: Fact Extraction in the Fast Lane. In *Proc. of the COLING-ACL06*, pages 809–816.

Patrick Pantel and Marco Pennacchiotti. 2006a. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proc. of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06*, pages 113–120.

Patrick Pantel and Pennacchiotti Pennacchiotti, Marco. 2006b. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proc. of the COLING-ACL06*, pages 113–120.

J. R. Quinlan. 1990. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266.

Matthew Richardson and Pedro Domingo. 2006. Markov logic networks. *Machine Learning*, 26:107–136.

Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *Proc. of EMNLP2010*, pages 1088–1098.

Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. 2008. TSUBAKI: An open search engine infrastructure for developing new information access. In *Proc. of IJCNLP*, pages 189–196.

Kentaro Torisawa, Stijn De Saeger, Jun'ichi Kazama, Asuka Sumida, Daisuke Noguchi, Yasunari Kakizawa, Masaaki Murata, Kow Kuroda, and Ichiro Yamada. 2010. Organizing the web's information explosion to discover unknown unknowns. *New Generation Computing*, 28(3):217–236.

Masaaki Tsuchida, Stijn De Saeger, Kentaro Torisawa, Masaki Murata, Jun'ichi Kazama, Kow Kuroda, and Hayato Ohwada. 2010. Large scale similarity-based relation expansion. In *Proc of the 4th IUCS*, pages 140–147.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, pages 1083–1106.

Identification of Multi-word Expressions by Combining Multiple Linguistic Information Sources

Yulia Tsvetkov
Language Technologies Institute
Carnegie Mellon University
yulia.tsvetkov@gmail.com

Shuly Wintner
Department of Computer Science
University of Haifa
shuly@cs.haifa.ac.il

Abstract

We propose an architecture for expressing various linguistically-motivated features that help identify multi-word expressions in natural language texts. The architecture combines various linguistically-motivated classification features in a Bayesian Network. We introduce novel ways for computing many of these features, and manually define linguistically-motivated interrelationships among them, which the Bayesian network models. Our methodology is almost entirely unsupervised and completely language-independent; it relies on few language resources and is thus suitable for a large number of languages. Furthermore, unlike much recent work, our approach can identify expressions of various types and syntactic constructions. We demonstrate a significant improvement in identification accuracy, compared with less sophisticated baselines.

1 Introduction

Multi-word Expressions (MWEs) are lexical items that consist of multiple orthographic words (e.g., *ad hoc*, *by and large*, *New York*, *kick the bucket*). MWEs are numerous and constitute a significant portion of the lexicon of any natural language (Jackendoff, 1997; Erman and Warren, 2000; Sag et al., 2002). They are a heterogeneous class of constructions with diverse sets of characteristics, distinguished by their idiosyncratic behavior. Morphologically, some MWEs allow some of their constituents to freely inflect while restricting (or preventing) the inflection of other constituents. In some cases MWEs may allow constituents to undergo non-standard morphological inflections that they would not undergo in isolation. Syntactically, some MWEs behave like words while other are phrases; some occur in one rigid pattern (and a fixed order), while others permit various syntactic transformations. Semantically, the compositionality of MWEs is gradual, ranging from fully compositional to idiomatic (Bannard et al., 2003).

Because of their prevalence and irregularity, MWEs must be stored in lexicons of natural language processing applications. Correct handling of MWEs has been proven beneficial for various applications, including information retrieval, building ontologies, text alignment, and machine translation.

We propose a novel architecture for identifying MWEs of various types and syntactic categories in monolingual corpora. Unlike much existing work, which focuses on a particular syntactic construction, our approach addresses MWEs of all types by focusing on the general idiosyncratic properties of MWEs rather than on specific properties of each sub-class thereof. While we only evaluate our methodology on bi-grams, it can in principle be extended to longer MWEs. The architecture uses Bayesian Networks (BN) to express multiple interdependent linguistically-motivated features.

First, we automatically generate a small (training) set of MWE and non-MWE bi-grams (positive and negative instances, respectively). We then define a set of linguistically-motivated features that embody observed characteristics of MWEs. We augment these by features that reflect collocation measures. Finally, we define dependencies among these features, expressed in the structure of a Bayesian Network model, which we then use for classification. This is a directed graph, whose nodes express the features used for classification, and whose edges de-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 836–845,
Edinburgh, Scotland, UK, July 27–31, 2011.

fine causal relationships among these features. In this architecture, learning does not result in a black box, expressed solely as feature weights. Rather, the structure of the BN allows us to learn the impact of different MWE features on the classification. The result is a new unsupervised method for identifying MWEs of various types in text corpora. It combines statistics with a large array of linguistically-motivated features, organized in an architecture that reflects interdependencies among the features.

The contribution of this work is manifold. First, we show how to generate training material (almost) automatically, so the method is almost completely unsupervised. The methodology we advocate is thus language-independent, requiring relatively few language resources, and is therefore optimal for medium-density languages (Varga et al., 2005). Second, we propose several linguistically-motivated features that can be computed from data and that are demonstrably productive for improving the accuracy of MWE identification. These feature focus on the expression of linguistic idiosyncrasies of various types, a phenomenon typical of MWEs. We propose novel computational modeling of many of these features; in particular, we account for the morphological idiosyncrasy of MWEs using a histogram of the number of inflected forms, in a technique that draws from image processing. Third, we advocate the use of Bayesian Networks as a mechanism for expressing manually-crafted dependencies among features; the use of BN significantly improves the classification accuracy. Finally, we demonstrate the utility of our methodology by applying it to Hebrew.[1] Our evaluation shows that the use of linguistically-motivated features results in reduction of 23% of the errors compared with a collocation baseline; organizing the knowledge in a BN reduces the error rate by additional 8.7%.

After discussing related work in the next section, we describe in Section 3 the methodology we propose, including a detailed discussion of the features and their implementation. Section 4 provides a thorough evaluation of the results. We conclude with suggestions for future research.

[1]To facilitate readability we use a transliteration of Hebrew using Roman characters; the letters used, in Hebrew lexicographic order, are *abgdhwzxTiklmns'pcqršt*.

2 Related Work

Early approaches to MWEs identification concentrated on their collocational behavior (Church and Hanks, 1990). Pecina (2008) compares 55 different association measures in ranking German Adj-N and PP-Verb collocation candidates. He shows that combining different collocation measures using standard statistical classification methods improves over using a single collocation measure. Other results (Chang et al., 2002; Villavicencio et al., 2007) suggest that some collocation measures (especially PMI and Log-likelihood) are superior to others for identifying MWEs.

Soon, however, it became clear that mere co-occurrence measurements are not enough to identify MWEs, and their linguistic properties should be exploited as well (Piao et al., 2005). Hybrid methods that combine word statistics with linguistic information exploit morphological, syntactic and semantic idiosyncrasies to extract idiomatic MWEs.

Ramisch et al. (2008) evaluate a number of association measures on the task of identifying English Verb-Particle Constructions and German Adjective-Noun pairs. They show that adding linguistic information (mostly POS and POS-sequence patterns) to the association measure yields a significant improvement in performance over using pure frequency.

Several works address the *lexical fixedness* or *syntactic fixedness* of (certain types of) MWEs in order to extract them from texts. An expression is considered lexically fixed if replacing any of its constituents by a semantically (and syntactically) similar word generally results in an invalid or literal expression. Syntactically fixed expressions prohibit (or restrict) syntactic variation. For example, Van de Cruys and Villada Moirón (2007) use lexical fixedness to extract Dutch Verb-Noun idiomatic combinations (VNICs). Bannard (2007) uses syntactic fixedness to identify English VNICs. Another work uses both the syntactic and the lexical fixedness of VNICs in order to distinguish them from non-idiomatic ones, and eventually to extract them from corpora (Fazly and Stevenson, 2006).

While these approaches are in line with ours, they require lexical semantic resources (e.g., a database that determines semantic similarity among words) and syntactic resources (parsers) that are unavail-

able for Hebrew (and many other languages). Our approach only requires morphological processing and a bilingual dictionary, which are more readily-available for several languages. Note also that these approaches target a specific syntactic construction, whereas ours is adequate for various types of MWEs.

Several properties of Hebrew MWEs are described by Al-Haj (2010); Al-Haj and Wintner (2010) use them in order to construct an SVM-based classifier that can distinguish between MWE and non-MWE *noun-noun constructions* in Hebrew. The features of the SVM reflect several morphological and morpho-syntactic properties of such constructions. The resulting classifier performs much better than a naïve baseline, reducing over one third of the errors. We rely on some of these insights, as we implement more of the linguistic properties of MWEs. Again, our methodology is not limited to a particular construction: indeed, we demonstrate that our general methodology, trained on automatically-generated, general training data, performs almost as well as the noun-noun-specific approach of Al-Haj and Wintner (2010) on the very same dataset.

Recently, Tsvetkov and Wintner (2010b) introduced a general methodology for extracting MWEs from bilingual corpora, and applied it to Hebrew. The results were a highly accurate set of Hebrew MWEs, of various types, along with their English translations. A major limitation of this work is that it can only be used to identify MWEs in the bilingual corpus, and is thus limited in its scope. We use this methodology to extract both positive and negative instances for our training set in the current work; but we extrapolate the results much further by extending the method to *mono*lingual corpora, which are typically much larger than bilingual ones.

Bayesian Networks have only scarcely been used for classification in natural language applications. For example, BN were used for POS tagging of unknown words (Peshkin et al., 2003); dependency parsing (Savova and Peshkin, 2005); and document classification (Lam et al., 1997; Calado et al., 2003; Denoyer and Gallinari, 2004). Very recently, Ramisch et al. (2010) have used BN for Portuguese MWE identification. The features used for classification were of two kinds: (1) various collocation measures; (2) bi-grams aligned together by an automatic word aligner applied to a parallel (Portuguese-English) corpus. A BN was used to combine the predictions of the various features on the test set, but the structure of the network is not described. The combined classifier resulted in a much higher accuracy than any of the two methods alone. However, the BN does not play any special role in this work, and its structure does not reflect any insights or intuitions on the structure of the problem domain or on interdependencies among features.

We, too, acknowledge the importance of combining different types of knowledge in the hard task of MWE identification. In particular, we also believe that collocation measures are highly important for this task, but cannot completely solve the problem: linguistically-motivated features are mandatory in order to improve the accuracy of the classifier. In this work we focus on various properties of different types of MWEs, and define general features that may accurately apply to some, but not necessarily all of them. An architecture of Bayesian Networks is optimal for this task: it enables us to define weighted dependencies among features, such that certain features are more significant for identifying some class of MWEs, whereas others are more prominent in identifying other classes. As we show below, this architecture results in significant improvements over a more naïve combination of features.

3 Methodology

3.1 Motivation

The task we address is identification of MWEs, of various types and syntactic constructions, in monolingual corpora.[2] Several properties of MWEs make this task challenging: MWEs exhibit idiosyncrasies on a variety of levels, orthographic, morphological, syntactic and of course semantic (Al-Haj, 2010). They are also extremely diverse: for example, on the semantic dimension alone, MWEs cover an entire spectrum, ranging from frozen, fixed idioms to free combinations of words (Bannard et al., 2003).

Such a complex task calls for a combination of multiple approaches, and much research indeed suggests "hybrid" approaches to MWE identification

[2]For simplicity, we focus on bi-grams of tokens (MWEs of length 2) in this work; the methodology, however, is easily extensible to longer n-grams.

(Duan et al., 2009; Weller and Fritzinger, 2010; Ramisch et al., 2010; Hazelbeck and Saito, 2010). We believe that Bayesian Networks provide an optimal architecture for expressing various pieces of knowledge aimed at MWE identification, for the following reasons (Heckerman, 1995):

- In contrast to many other classification methods, BN can learn (and express) causal relationships between features. This facilitates better understanding of the problem domain.
- BN can encode not only statistical data, but also prior domain knowledge and human intuitions, in the form of interdependencies among features. We do indeed use this possibility here.

3.2 Linguistically-motivated Features

Based on the observations of Al-Haj (2010), we define several linguistically-motivated features that are aimed at capturing some of the unique properties of MWEs. While many idiosyncratic properties of MWEs have been previously studied, we introduce novel ways to express those properties as computable features informing a classifier. Note that many of the features we describe below are completely language-independent; others are applicable to a wide range of languages, while few are specific to morphologically-rich languages, and can be exhibited in different ways in different languages. The methodology we advocate, however, is completely universal.

A common theme for all these features is *idiosyncracy*: they are all aimed at locating some linguistic property on which MWEs may differ from non-MWEs. Below we detail these properties, along with the features that we define to reflect them. In all cases, the feature is applied to a *candidate MWE*, defined here as a bi-gram of tokens (all possible bigrams are potential candidates). To compute the features, we use a 46M-token monolingual Hebrew corpus (Itai and Wintner, 2008), which we pre-process as in Tsvetkov and Wintner (2010b). All statistics are computed from this large corpus. Likewise, we compute these features on a small training corpus, which we generate automatically (see Section 3.4).

Orthographic variation Sometimes, MWEs are written with dashes instead of inter-token spaces. We define a binary feature, DASH, whose value is 1 iff the dash character appears in some surface form of the candidate MWE. For example, *xd-cddi* (*one sided*) "unilateral".

Hapax legomena MWEs sometimes include constituents that have no usage outside the particular expression, and are hence not included in lexicons. We define a feature, HAPAX, whose value is a binary vector with 1 in the i-th place iff the i-th word of the candidate is not in the lexicon, and does not occur in other bi-grams at the same location. For example, *hwqws pwqws* "hocus-pocus". In order to filter out potential errors, candidates must occur at least 5 times in the corpus in order for this feature to fire.

Frozen form MWE constituents sometimes occur in one fixed, frozen form. We define a feature, FROZEN, whose value is a binary vector with 1 in the i-th place iff the i-th word of the candidate never inflects in the context of this expression. Example: *bit xwlim* (*house-of sick-people*) "hospital"; the noun *xwlim* must be in the plural in this MWE.

Partial morphological inflection In some cases, MWE constituents undergo a (strict but non-empty) subset of the full inflections that they would undergo in isolation. We capture this property with a technique that has been proven useful in the area of image processing (Jain, 1989, Section 7.3). We compute a histogram of the distribution in the corpus of all the possible surface forms of each constituent of an MWE candidate. Such histograms can compactly represent distributional information on morphological behavior, in the same way that histograms of the distribution of gray levels in a picture are used to represent the picture itself.

Our assumption is that the inflection histograms of non-MWEs are more uniform than the histograms of MWEs, in which some inflections may be more frequent and others may be altogether missing. Of course, restrictions on the histogram may stem from the part of speech of the expression; such constraints are captured by dependencies in the BN structure.

Since each MWE is idiosyncratic in its own way, we do not expect the histograms of MWEs to have some specific pattern, except non-uniformity. We therefore sort the columns of each histogram, thereby losing information pertaining to the specific

inflections, and retaining only information about the idiosyncrasy of the histogram. Offline, we compute the average histogram for positive and negative examples: The average histogram of MWEs is shorter and less uniform than the average histogram of non-MWEs. We define as feature, HIST, the L_1 (Manhattan) distance between the histogram of the candidate and the closest average histogram.

For example, the MWE *bit mepv* (*house-of law*) "court" occurs in the following inflected forms: *bit hmepv* "the court" (75%); *bit mepv* "a court" (15%); *bti hmepv* "the courts" (8%); and *bti mepv* "courts" (2%). The histogram for this candidate is thus $(75, 15, 8, 2)$. In contrast, the non-MWE *txwm mepv* (*domain-of law*) "domain of the law", which is syntactically identical, occurs in nine different inflected forms, and its sorted histogram is $(59, 14, 7, 7, 5, 2, 2, 2, 2)$.

Context We hypothesize that MWEs tend to constrain their (semantic) context more strongly than non-MWEs. We expect words that occur immediately after MWEs to vary less freely than words that immediately follow other expressions. One motivation for this hypothesis is the observation that MWEs tend to be less polysemous than free combinations of words, thereby limiting the possible semantic context in which they can occur.

We define a feature, CONTEXT, as follows. We first compute a histogram of the frequencies of words following each candidate MWE. We trim the tail of the histogram by removing words whose frequency is lower than 0.1% (the expectation is that non-MWEs would have a much longer tail). Offline, we compute the same histograms for positive and negative examples and average them as above. The value of CONTEXT is 1 iff the histogram of the candidate is closer (in terms of L_1 distance) to the positive average.

For example, the histogram of *bit mepv* "court" includes 15 values, dominated by *bit mepv yliwn* "supreme court" (20%) and *bit mepv mxwzi* "district court" (13%), followed by contexts whose frequency ranges between 5% and 0.6%. In contrast, the non-MWE *txwm mepv* "domain-of law" has a much shorter histogram, namely $(12, 11, 6)$: over 70% of the words following this expression occur less than 0.1% and are hence in the trimmed tail.

Syntactic diversity MWEs can belong to various part of speech categories. We define as feature, POS, the category of the candidate, with values obtained by selecting frequent tuples of POS tags. For example, Noun-Noun, PropN-PropN, Noun-Adj, etc.

Translational equivalents Since MWEs are often idiomatic, they tend to be translated in a non-literal way, sometimes to a single word. We use a dictionary to generate word-by-word translations of candidate MWEs to English, and check the number of occurrences of the English literal translation in a large English corpus.[3] Due to differences in word order between the two languages, we create two variants for each translation, corresponding to both possible orders. We expect non-MWEs to have some literal translational equivalent (possibly with frequency that correlates with their frequency in Hebrew), whereas for MWEs we expect no (or few) literal translations. We define a binary feature, TRANS, whose value is 1 iff some literal translation of the candidate occurs more than 5 times in the corpus.

For example, the MWE *htxtn ym* (*marry with*) "marry" is literally translated as *with marry*, *marry with*, *together marry* and *marry together*, none of which occurs in the corpus.

Collocation As a baseline, statistical association measure, we use a heuristic variant of pointwise mutual information (PMI), promoting also collocations whose constituents are frequent (Tsvetkov and Wintner, 2010b). We define a binary feature, PMI, with values (*low* and *high*) reflecting the threshold that maximizes the accuracy of MWE classification in Tsvetkov and Wintner (2010b).

3.3 Feature Interdependencies Expressed as a Bayesian Network

A Bayesian Network (Jensen and Nielsen, 2007) is organized as a graph whose nodes are random variables and whose edges represent interdependencies among those variables. We use a particular type of BN, known as *causal* networks, in which directed edges lead to a variable from each of its direct *causes*. This facilitates the expression of domain knowledge (and intuitions, beliefs, etc.) as structural properties of the network. We use the BN as

[3] We use a 120M-token newspaper corpus.

a classification device: training amounts to computing the joint probability distribution of the training set, whereas classification maximizes the posterior probability of the particular node (variable) being queried.

For MWE identification we define a BN whose nodes correspond to the features described in Section 3.2. In addition, we define a node MWE for the complete classification task. Over these nodes we impose the structure depicted graphically in Figure 1. This structure, which we motivate below, is *manually* defined: it reflects our understanding of the problem domain and is a result of thorough experimentations. That said, it can of course be modified in various ways, and in particular, new nodes can be easily added to reflect additional features.

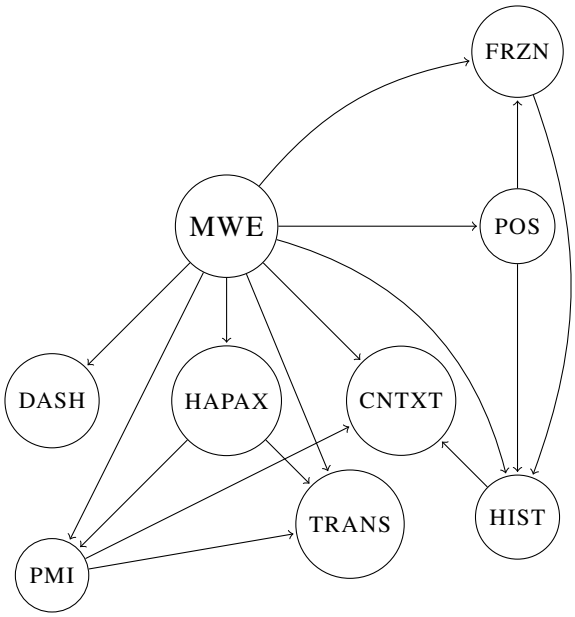


Figure 1: Bayesian Network for MWE identification

All nodes depend on MWE, as all are affected by whether or not the candidate is a MWE. The POS of an expression influences its morphological inflection, hence the edges from POS to HIST and to FROZEN. For example, Hebrew noun-noun constructions allow their constituents to undergo the full inflectional paradigm, but when such a construction is a MWE, inflection is severely constrained (Al-Haj and Wintner, 2010); similarly, when one of the constituents of a MWE is a conjunction, the entire expression is very likely to be frozen.

Hapaxes clearly affect all statistical metrics, hence the edge from HAPAX to PMI, and also the existence of literal translation, since if a word is not in the lexicon, it does not have a translation, hence the edge from HAPAX to TRANS. Also, we assume that there is a correlation between the frequency (and PMI) of a candidate and whether or not a literal translation of the expression exists, hence the edge from PMI to TRANS. The edges from PMI and HIST to CONTEXT are justified by the correlation between the frequency and variability of an expression and the variability of the context in which it occurs.

Once the structure of the network is established, the conditional probabilities of each dependency have to be determined. We compute the conditional probability tables from our training data (see below) using Weka (Hall et al., 2009), and obtain values for $P(X \mid X_1, \ldots, X_k)$ for each variable X and all variables X_i, $1 \leq i \leq k$, such that the graph includes an edge from X_i to X (parents of X). We then perform inference on the network in order to compute $P(X_{mwe} \mid X_1, \ldots, X_k)$, where X_{mwe} corresponds to the node MWE, and $X_1, \ldots, X_k$ are the variables corresponding to all *other* nodes in the network. Using Bayes Rule,

$$P(X_{mwe} \mid X_1, \ldots, X_k) \propto$$
$$P(X_1, \ldots, X_k \mid X_{mwe}) \times P(X_{mwe})$$

We define the prior, $P(X_{mwe})$, to be 0.41: this is the percentage of MWEs in WordNet 1.7 (Fellbaum, 1998). The conditional probabilities $P(X_1, \ldots, X_k \mid X_{mwe})$ are determined by Weka from the conditional probability tables:

$$P(X_1, \ldots, X_k \mid X_{mwe}) = \Pi_{i=1}^{k} P(X_i \mid \mathbf{pa}_i)$$

where k is the number of nodes in the BN (other than X_{mwe}) and $\mathbf{pa}_i$ is the set of parents of X_i.

3.4 Automatic Generation of Training Data

For training we need samples of positive and negative instances of MWEs, each associated with a vector of the values of all features discussed in Section 3.2. We generate this training material automatically. We use a small Hebrew-English bilingual corpus (Tsvetkov and Wintner, 2010a). We word-align the corpus with Giza++ (Och and Ney, 2003), and then apply the (completely unsupervised)

algorithm of Tsvetkov and Wintner (2010b), which extracts MWE candidates from the aligned corpus and re-ranks them using statistics computed from a large monolingual corpus. The core idea behind this method is that MWEs tend to be translated in non-literal ways; in a parallel corpus, words that are 1:1 aligned typically indicate literal translations and are hence unlikely constituents of MWEs.

The result is a set of 134,001 Hebrew bi-gram types (from the bilingual corpus), classified as either 1:1 aligned (implying they are likely *not* MWEs) or unaligned (in which case they may or may not be MWEs). In addition, for each bi-gram we have a PMI score; naturally, higher PMI scores are indicative of MWEs. We thus divide the set into four classes: aligned bi-grams with high PMI score, aligned bi-grams with low PMI score, misaligned with high PMI and misaligned with low PMI. Aligned bi-grams, independently of their PMI score, are more likely non-MWEs; high-PMI misaligned bi-grams are very likely MWEs; and the status of low-PMI misaligned bi-grams is unclear, and must be further investigated. This is summarized in Table 1.

	Misaligned	Aligned
High PMI	MWE	non-MWE
Low PMI	unclear	non-MWE

Table 1: Classification of bi-grams

We set the threshold that separates low PMI from high PMI as in Tsvetkov and Wintner (2010b). The results of this classification is depicted in Table 2.

	Misaligned	Aligned	Total
High PMI	2,203	493	2,696
Low PMI	61,314	69,991	131,305
Total	63,517	70,484	134,001

Table 2: Statistics of the sample space from which the training set is generated

We assume that all bi-grams in the 'Aligned' column are non-MWEs. Additionally, we assume that the 2,203 misaligned bi-grams with high PMI scores are likely MWEs. As for the set of over 61,000 misaligned low-PMI bi-grams, certainly many of them are non-MWEs, but some may be MWEs, and we are interested in including them as positive examples of MWEs with low PMI scores. We therefore manually annotate a sample of 50 MWEs from this particular set (we had to manually go over a few thousands of bi-grams to select this sample). This is the only supervision provided in this work.

The remaining question is how to determine the sizes of samples from each of the other three classes. We use two guidelines: first, we would like the ratio of MWEs to non-MWEs in the training set to be $41 : 59$, reflecting the ratio in WordNet (the prior MWE probability). Second, we would like classification by PMI score only to yield a reasonable baseline; the baseline is defined as the ratio of the sum of high-PMI MWEs plus low-PMI non-MWEs to the size of the training set. We choose 67%, the PMI baseline reported by Al-Haj and Wintner (2010). As a result of these two considerations, we end up with training sets whose sizes are depicted in Table 3. We randomly select from the sample space this many instances for each class. Since much of the procedure of preparing training data is automatic, the results may be somewhat noisy. As Bayesian Network are known to be robust to noisy data, we expect the BN to compensate for this problem.

	MWE	non-MWE	Total
High PMI	300	232	532
Low PMI	50	272	322
Total	350	504	854

Table 3: Sizes of each training set

4 Results and Evaluation

We use the training set described above for training and evaluation: we perform 10-fold cross validation experiments, reporting Precision, Recall, Accuracy and F-measure in three setups: one (SVM) in which we train an SVM classifier[4] with the features described in Section 3.2; one (BN-auto) in which we train a BN but let Weka determine its structure (using the K2 algorithm); and one (BN) in which we train a Bayesian Network whose structure reflects manually-crafted linguistically-motivated knowledge, as depicted in Figure 1. The

[4]We use Weka SMO with the PolyKernel setup; experimentation with several other kernels yielded worse results.

results, along with the PMI baseline figures, are listed in Table 4.

	Accuracy	Prec.	Recall	F-score
PMI	66.98%	0.73	0.67	0.67
BN-auto	71.19%	0.71	0.71	0.71
SVM	74.59%	0.75	0.75	0.75
BN	**76.82%**	**0.77**	**0.77**	**0.77**

Table 4: 10-fold cross validation evaluation results

The linguistically-motivated features defined in Section 3.2 are clearly helpful in the classification task: the accuracy of the SVM, informed by these features, is close to 75%, reducing the error rate of the PMI baseline by 23%. The contribution of the Bayesian Network is also highly significant, reducing almost 7% more errors (8.7% of the errors made by the SVM classifier), or a total of almost 30% error-rate reduction with respect to the baseline. Interestingly, a BN whose structure does not reflect prior knowledge, but is rather learned automatically, performs poorly. It is the combination of linguistically-motivated features with feature interdependencies reflecting domain knowledge that contribute to the best performance.

As a further demonstration of the utility of our approach, we evaluate the algorithm on an additional test set that was used for evaluation in the past (Tsvetkov and Wintner, 2010b; Al-Haj and Wintner, 2010). This is a small annotated corpus, **NN**, of Hebrew noun-noun constructions. The corpus consists of 413 high-frequency bi-grams of the same syntactic construction; of those, 178 are tagged as MWEs (in this case, noun compounds) and 235 as non-MWEs. This corpus consolidates the annotation of three annotators: only instances on which all three agreed were included. Since it includes both positive and negative instances, this corpus facilitates a robust evaluation of precision and recall.

We train a Bayesian Network on the training set described in Section 3.4 and use it to classify the set **NN**. We compare the results of this classifier with a PMI baseline (using the same threshold as above), and also with the classification results reported by Al-Haj and Wintner (2010) (AW); the latter reflects 10-fold cross-validation evaluation using the entire set, so it should be considered an upper bound for any classifier that uses a general training corpus.

The results are depicted in Table 5. They clearly demonstrate that the linguistically-motivated features we define provide a significant improvement in classification accuracy over the baseline PMI measure. Note that our F-score, 0.77, is very close to the best result of 0.79 obtained by Al-Haj and Wintner (2010) as the average of 10-fold cross validation runs, using *only* high-frequency noun-noun constructions for training. We interpret this result as a further proof of the robustness of our architecture.

	Accuracy	Precision	Recall	F-score
PMI	71.43%	0.71	0.71	0.71
BN	**77.00%**	**0.77**	**0.77**	**0.77**
AW	80.77%	0.77	0.81	0.79

Table 5: Evaluation results: noun-noun constructions

Finally, we have used the trained BN to classify the entire set of bi-grams present in the (Hebrew side of the) parallel corpus described in Tsvetkov and Wintner (2010a). Of the 134,000 candidates, only 4,000 are classified as MWEs. We sort this list of potential MWEs by the probability assigned by the BN to the positive value of the variable X_{mwe}. The resulting sorted list is dominated by high-PMI bi-grams, especially proper names, all of which are indeed MWEs. The first non-MWE (false positive) occurs in the 50th place on the list; it is *crpt niqwla* "France Nicolas", which is obviously a sub-sequence of the larger MWE, *neia crpt niqwla srqwzi* "French president Nicolas Sarkozy". Similar sub-sequences are also present, but only five are in the top-100. Such false positives can be reduced when longer MWEs are extracted, as it can be assumed that a sub-sequence of a longer MWE does not have to be identified. Other false positives in the top-100 include some highly frequent expressions, but over 85 of the top-100 are clearly MWEs.

While more careful evaluation is required in order to estimate the rate of true positives in this list, we trust that the vast majority of the positive results are indeed MWEs.

5 Conclusions and future work

We presented a novel architecture for identifying MWEs in text corpora. The main insights we em-

phasize are sophisticated computational encoding of linguistic knowledge that focuses on the idiosyncratic behavior of such expressions. This is reflected in two ways in our work: by defining computable features that reflect different facets of irregularities; and by framing the features as part of a larger Bayesian Network that accounts for interdependencies among them. We also introduce a method for automatically generating a training set for this task, which renders the classification almost entirely unsupervised. The result is a nearly-unsupervised, language-independent classification method that can identify MWEs of various lengths, types and constructions. Evaluation on Hebrew shows significant improvement in the accuracy of the classifier compared with the state of the art.

The modular architecture of BN facilitates easy exploration with more features. We are currently investigating the contribution of various other sources of information to the classification task. For example, Hebrew lacks large-scale lexical semantic resources. However, it is possible to literally translate a MWE candidate to English and rely on the English WordNet for generating synonyms of the literal translation. Such "literal synonyms" can then be back-translated to Hebrew. The assumption is that if a back-translated expression has a high PMI, the original candidate is very likely not a MWE. While such a feature may contribute little on its own, incorporating it in a well-structured BN may improve performance.

While our methodology is applicable to MWEs of any length, we have so far only evaluated it on bigrams. In the future, we intend to extend the evaluation to longer n-grams. We also plan to apply the methodology to languages other than Hebrew.

Acknowledgments

This research was supported by THE ISRAEL SCIENCE FOUNDATION (grants No. 137/06, 1269/07). We are grateful to Gennadi Lembersky for his continuous help.

References

Hassan Al-Haj and Shuly Wintner. 2010. Identifying multi-word expressions by leveraging morphological and syntactic idiosyncrasy. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 10–18, Beijing, China, August. Coling 2010 Organizing Committee.

Hassan Al-Haj. 2010. Hebrew multiword expressions: Linguistic properties, lexical representation, morphological processing, and automatic acquisition. Master's thesis, University of Haifa, February.

Colin Bannard, Timothy Baldwin, and Alex Lascarides. 2003. A statistical approach to the semantics of verb-particles. In Diana McCarthy Francis Bond, Anna Korhonen and Aline Villavicencio, editors, *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 65–72.

Colin Bannard. 2007. A measure of syntactic flexibility for automatically identifying multiword expressions in corpora. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 1–8. Association for Computational Linguistics.

Pável Calado, Marco Cristo, Edleno Silva De Moura, Nivio Ziviani, Berthier A. Ribeiro-Neto, and Marcos André Gonçalves. 2003. Combining link-based and content-based methods for web document classification. In *Proceedings of CIKM-03, 12th ACM International Conference on Information and Knowledge Management*, pages 394–401, New Orleans, US. ACM Press, New York, US.

Baobao Chang, Pernilla Danielsson, and Wolfgang Teubert. 2002. Extraction of translation unit from Chinese-English parallel corpora. In *Proceedings of the first SIGHAN workshop on Chinese language processing*, pages 1–5, Morristown, NJ, USA. Association for Computational Linguistics.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Ludovic Denoyer and Patrick Gallinari. 2004. Bayesian network model for semi-structured document classification. *Information Processing and Management*, 40(5):807–827.

Jianyong Duan, Mei Zhang, Lijing Tong, and Feng Guo. 2009. A hybrid approach to improve bilingual multiword expression extraction. In Thanaruk Theeramunkong, Boonserm Kijsirikul, Nick Cercone, and Tu-Bao Ho, editors, *Advances in Knowledge Discovery and Data Mining*, volume 5476 of *Lecture Notes in Computer Science*, pages 541–547. Springer, Berlin and Heidelberg.

Britt Erman and Beatrice Warren. 2000. The idiom principle and the open choice principle. *Text*, 20(1):29–62.

Afsaneh Fazly and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 337–344.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. MIT Press.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

Gregory Hazelbeck and Hiroaki Saito. 2010. A hybrid approach for functional expression identification in a japanese reading assistant. In *Proceedings of the 2010 Workshop on Multiword Expressions: from Theory to Applications*, pages 81–84, Beijing, China, August. Coling 2010 Organizing Committee.

David Heckerman. 1995. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March.

Alon Itai and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42(1):75–98, March.

Ray Jackendoff. 1997. *The Architecture of the Language Faculty*. MIT Press, Cambridge, USA.

Anil K. Jain. 1989. *Fundamentals of digital image processing*. Prentice-Hall, Inc., NJ, USA.

Finn V. Jensen and Thomas D. Nielsen. 2007. *Bayesian Networks and Decision Graphs*. Springer, 2nd edition.

Wai Lam, Kon F. Low, and Chao Y. Ho. 1997. Using a bayesian network induction approach for text categorization. In Martha E. Pollack, editor, *Proceedings of IJCAI-97, 15th International Joint Conference on Artificial Intelligence*, pages 745–750, Nagoya, JP. Morgan Kaufmann Publishers, San Francisco, US.

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

Pavel Pecina. 2008. A machine learning approach to multiword expression extraction. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions*.

Leonid Peshkin, Avi Pfeffer, and Virginia Savova. 2003. Bayesian nets in syntactic categorization of novel words. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers - Volume 2*, NAACL '03, pages 79–81, Morristown, NJ, USA. Association for Computational Linguistics.

Scott Songlin Piao, Paul Rayson, Dawn Archer, and Tony McEnery. 2005. Comparing and combining a semantic tagger and a statistical tool for mwe extraction. *Computer Speech and Language*, 19(4):378–397.

Carlos Ramisch, Paulo Schreiner, Marco Idiart, and Alline Villavicencio. 2008. An evaluation of methods for the extraction of multiword expressions. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions*.

Carlos Ramisch, Helena de Medeiros Caseli, Aline Villavicencio, André Machado, and Maria Finatto. 2010. A hybrid approach for multiword expression identification. In Thiago Pardo, António Branco, Aldebaro Klautau, Renata Vieira, and Vera de Lima, editors, *Computational Processing of the Portuguese Language*, volume 6001 of *Lecture Notes in Computer Science*, pages 65–74. Springer.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2002)*, pages 1–15, Mexico City, Mexico.

Virginia Savova and Leonid Peshkin. 2005. Dependency parsing with dynamic bayesian network. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 3*, pages 1112–1117. AAAI Press.

Yulia Tsvetkov and Shuly Wintner. 2010a. Automatic acquisition of parallel corpora from websites with dynamic content. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, pages 3389–3392. European Language Resources Association (ELRA), May.

Yulia Tsvetkov and Shuly Wintner. 2010b. Extraction of multi-word expressions from small parallel corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, August.

Tim Van de Cruys and Begoña Villada Moirón. 2007. Semantics-based multiword expression extraction. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 25–32, Prague, Czech Republic, June. Association for Computational Linguistics.

Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2005. Parallel corpora for medium density languages. In *Proceedings of RANLP'2005*, pages 590–596.

Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart, and Carlos Ramisch. 2007. Validation and evaluation of automatically acquired multiword expressions for grammar engineering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1034–1043.

Marion Weller and Fabienne Fritzinger. 2010. A hybrid approach for the identification of multiword expressions. In *Proceedings of the SLTC 2010 Workshop on Compounds and Multiword Expressions*, October.

Analyzing Methods for Improving Precision of Pivot Based Bilingual Dictionaries

Xabier Saralegi, Iker Manterola, Iñaki San Vicente
R&D Elhuyar Foundation
Zelai haundi 3, Osinalde Industrialdea
20170 Usurbil, Basque Country
{x.saralegi, i.manterola, i.sanvicente}@elhuyar.com

Abstract

An A-C bilingual dictionary can be inferred by merging A-B and B-C dictionaries using B as pivot. However, polysemous pivot words often produce wrong translation candidates. This paper analyzes two methods for pruning wrong candidates: one based on exploiting the structure of the source dictionaries, and the other based on distributional similarity computed from comparable corpora. As both methods depend exclusively on easily available resources, they are well suited to less resourced languages. We studied whether these two techniques complement each other given that they are based on different paradigms. We also researched combining them by looking for the best adequacy depending on various application scenarios.

1 Introduction

Nobody doubts the usefulness and multiple applications of bilingual dictionaries: as the final product in lexicography, translation, language learning, etc. or as a basic resource in several fields such as Natural Language Processing (NLP) or Information Retrieval (IR), too. Unfortunately, only major languages have many bilingual dictionaries. Furthermore, construction by hand is a very tedious job. Therefore, less resourced languages (as well as less-common language pairs) could benefit from a method to reduce the costs of constructing bilingual dictionaries. With the growth of the web, resources like Wikipedia seem to be a good option to extract new bilingual lexicon (Erdmann et al., 2008), but the reality is that a dictionary is quite different from an encyclopedia. Wiktionary[1] is a promising asset more oriented towards lexicography. However, the presence of less resourced languages in these kinds of resources is still relative -in Wikipedia, too-.

Another way to create bilingual dictionaries is by using the most widespread languages (e.g., English, Spanish, French...) as a bridge between less resourced languages, since most languages have some bilingual dictionary to/from a major language. These pivot techniques allow new bilingual dictionaries to be built automatically. However, as the next section will show, it is no small task because translation between words is not a transitive relation at all. The presence of polysemous or ambiguous words in any of the dictionaries involved may produce wrong translation pairs. Several techniques have been proposed to deal with these ambiguity cases (Tanaka and Umemura, 1994; Shirai and Yamamoto, 2001; Bond et al., 2001; Paik et al., 2004; Kaji et al., 2008; Shezaf and Rappoport, 2010). However, each technique has different performance and properties producing dictionaries of certain characteristics, such as different levels of coverage of entries and/or translations. The importance of these characteristics depends on the context of use of the dictionary. For example, a small dictionary containing the most basic vocabulary and the corresponding most frequent translations can be adequate for some IR and NLP tasks, tourism, or initial stages of language learning. Alternatively, a dictionary which maximizes the vocabulary coverage is more oriented towards advanced users or translation services.

This paper addresses the problem of pruning

[1] http://www.wiktionary.org/

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 846–856,
Edinburgh, Scotland, UK, July 27–31, 2011.

wrong translations when building bilingual dictionaries by means of pivot techniques. We aimed to come up with a method suitable for less resourced languages. We analyzed two of the approaches proposed in the literature which are not very demanding on resources: Inverse Consultation (IC) (Tanaka and Umemura, 1994) and Distributional Similarity (DS) (Kaji et al., 2008), their strong points and weaknesses, and proposed that these two paradigms be combined. For this purpose, we studied the effect the attributes of the source dictionaries have on the performance of IC and DS-based methods, as well as the characteristics of the dictionaries produced. This could allow us to predict the performance of each method just by looking at the characteristics of the source dictionaries. Finally, we tried to provide the best combination adapted to various application scenarios which can be extrapolated to other languages.

The basis of the pivot technique is dealt with in the next section, and the state of the art in pivot techniques is reviewed in the third section. After that, the analysis of the aforementioned approaches and experiments carried out for that purpose are presented, and a proposal for combining both paradigms is included. The paper ends by drawing some conclusions from the results.

2 Pivot Technique

The basic pivot-oriented construction method is based on assuming the transitive relation of the translation of a word between two languages. Thus:

if p (pivot word) is a translation of s (source word) in the A-B dictionary and t (target word) is a translation of p in the B-C dictionary, we can say that t is therefore a translation of s, or $translation_{A,B}(s) = p$ and $translation_{B,C}(p) = t \rightarrow translation_{A,C}(s) = t$

This simplification is incorrect because it does not take into account word senses. Translations correspond to certain senses of the source words. If we look at figure 1, t (case of t_1 and t_2) can be the translation of p (p_2) for a sense c (c_3) different from the sense for which p (p_2) is the equivalent of s (c_1). This can happen when p pivot word is polysemous.

It could be thought that these causalities are not frequent, and that the performance of this basic approach could be acceptable. Let us analyze a real case. We merged a Basque-English dictionary composed of 17,672 entries and 43,021 pairs with an English-Spanish one composed of 16,326 entries and 38,128 pairs, and obtained a noised Basque-Spanish dictionary comprising 14,000 entries and 104,165 pairs. 10,000 (99,844 pairs) among all the entries have more than one translation. An automatic evaluation shows that 80.32% of these ambiguous entries contain incorrect translation equivalents (80,200 pairs out of 99,844). These results show that a basic pivot-oriented method is very sensitive to the ambiguity level of the source dictionaries. The conclusion is that the transitive relation between words across languages can not be assumed, because of the large number of ambiguous entries that dictionaries actually have. A more precise statement for the transitive property in the translation process would be:

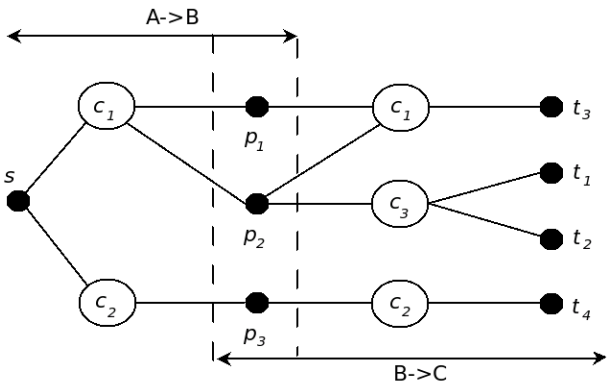


Figure 1: Ambiguity problem of the pivot technique.

if p (pivot word) is a translation of s with respect to a sense c and t is a translation of p with respect to the same sense c we can say that t is a translation of s, or $translation_{A,B}(s_{c_1}) = p$ and $translation_{B,C}(p_{c_2}) = t$ and $c_1 = c_2 \rightarrow translation_{A,C}(s) = t$

Unfortunately, most dictionaries lack comparable information about senses in their entries. So it is not possible to map entries and translation equivalents according to their corresponding senses. As an alternative, most papers try to guide this mapping according to semantic distances extracted from the dictionaries themselves or from external resources

such as corpora.

Another problem inherent in pivot-based techniques consists of missing translations. This consists of pairs of equivalents not identified in the pivot process because there is no pivot word, or else one of the equivalents is not present. We will not be dealing with this issue in this work so that we can focus on the translation ambiguity problem.

3 State of the Art

In order to reject wrong translation pairs, Tanaka et al. (1994) worked with the structure of the source dictionaries and introduced the IC method which measures the semantic distance between two words according to the number of pivot-words they share. This method was extended by using additional information from dictionaries, such as semantic classes and POS information in (Bond et al., 2001; Bond and Ogura, 2007). Sjöbergh (2005) compared full definitions in order to detect words corresponding to the same sense. However, not all the dictionaries provide this kind of information. Therefore, external knowledge needs to be used in order to guide mapping according to sense. István et al. (2009) proposed using WordNet, only for the pivot language (for English in their case), to take advantage of all the semantic information that WordNet can provide. Mausam et. al. (2009) researched the use of multiple languages as pivots, on the hypothesis that the more languages used, the more evidences will be found to find translation equivalents. They used Wiktionary for building a multilingual lexicon. Tsunakawa et al. (2008) used parallel corpora to estimate translation probabilities between possible translation pairs. Those reaching a minimum threshold are accepted as correct translations to be included in the target dictionary. However, even if this strategy achieves the best results in the terminology extraction field, it is not adequate when less resourced languages are involved because parallel corpora are very scarce.

As an alternative, (Kaji et al., 2008; Gamallo and Pichel, 2010) proposed methods to eliminate spurious translations using cross-lingual context or distributional similarity calculated from comparable corpora. In this line of work, (Shezaf and Rappoport, 2010) propose a variant of DS, and show how it outperforms the IC method. In comparison, our work focuses on analyzing the strong and weak points of each technique and aims to combine the benefits of each of them.

Other characteristics of the merged dictionaries like directionality (Paik et al., 2004) also influence the results.

4 Experimental Setup

This work focuses on adequate approaches for less resourced languages. Thus, the assumption for the experimentation is that few resources are available for both source and target languages. The resources for building the new dictionary are two basic (no definitions, no senses) bilingual dictionaries (A-B, B-C) including source (A), target (C) and a pivot language (B), as well as a comparable corpus for the source-target (A-C) language pair. We explored the IC (Tanaka and Umemura, 1994) and DS (Kaji et al., 2008; Gamallo and Pichel, 2010) approaches. In our experiments, the source and target languages are Basque and Spanish, respectively, and English is used for pivot purposes. In any case, the experiments could be conducted with any other language set, so long the required resources are available.

It must be noted that the proposed task is not a real problem because there is a Basque-Spanish dictionary already available. Resources like parallel corpora for that language pair are also available. These dictionaries and pivot language were selected in order to be able to evaluate the results automatically. During the evaluation we also used frequency information extracted from a parallel corpus, but then again, this corpus was not used during the dictionary building process, and therefore, it would not be used in a real application environment.

4.1 Resources

In order to carry out the experiments we used three dictionaries. The two dictionaries mentioned in the previous section (Basque-English $D_{eu \rightarrow en}$ and English-Spanish $D_{en \rightarrow es}$) were used to produce a new Basque-Spanish $D_{eu \rightarrow en \rightarrow es}$ dictionary. In addition, we used a Basque-Spanish $D_{eu \rightarrow es}$ dictionary for evaluation purposes. Its broad coverage is indicative of its suitability as a reference

dictionary. Table 1 shows the main characteristics of the dictionaries. We can observe that the ambiguity level of the entries (average number of translations per source word) is significant. This produces more noise in the pivot process, but it also benefits IC due to the increase in pivot words. As for the directions of source dictionaries, English is taken as target. Like Paik et al. (2004) we obtained the best coverage of pairs in that way.

Dictionary	**#entries**	**#pairs**	**ambiguity level**
$D_{eu \to en}$	17,672	43,021	2.43
$D_{en \to es}$	16,326	38,128	2.33
$D_{eu \to es}$(reference)	57,334	138,579	2.42
$D_{eu \to en \to es}$(noisy)	14,601	104,172	7.13

Table 1: Characteristics of the dictionaries.

Since we were aiming to merge two general dictionaries, the most adequate strategy was to use open domain corpora to compute DS. The domain of journalism is considered to be close to the open domain, and so we constructed a Basque-Spanish comparable corpus composed of news articles (see Table 2). The articles were gathered from the newspaper Diario Vasco (Hereinafter DV) for the Spanish part and from the Berria newspaper for the Basque part. Both publications focus on the Basque Country. In order to achieve a higher comparability degree, some constraints were applied:

- News in both languages corresponded to the same time span, 2006-2010.
- News corresponding to unrelated categories between newspapers were discarded.

Corpus	**#words**	**#docs**
Berria(eu)	40Mw	149,892
DV(es)	77Mw	306,924

Table 2: Characteristics of the comparable corpora.

In addition, as mentioned above, we extracted the frequencies of translation pairs from a Basque-Spanish parallel corpus. The corpus had 295,026 bilingual segments (4 Mw in Basque and 4.7 Mw in Spanish) from the domain of journalism.

5 Pruning Methods

IC and DS a priori suffer different weak points. IC depends on the structure of the source dictionaries. On the other hand, DS depends on a good comparable corpus and translation process. DS is measured more precisely between frequent words because context representation is richer.

The conditions for good performance of both IC and DS are analyzed below. These conditions will then be linked to the required characteristics for the initial dictionaries. In addition, we will measure how divergent the entries solved for each method are.

5.1 Inverse consultation

IC uses the structure of the D_{a-b} and D_{b-c} source dictionaries to measure the similarity of the meanings between source word and translation candidate. The description provided by Tanaka et al. (1994) is summarized as follows. To find suitable equivalents for a given entry, all target language translations of each pivot translation are looked up (e.g., $D_{b \to c}(D_{a \to b}(s))$). This way, all the "equivalence candidates" (ECs) are obtained. Then, each one is looked up in the inverse direction (following the previous example, $D_{c \to b}(t)$) to create a set of words called "selection area" (SA). The number of common elements of the same language between SA and the translations or equivalences (E) obtained in the original direction ($D_{a \to b}(s)$) is used to measure the semantic distance between entries and corresponding translations. The more matches there are, the better the candidate is. If only one inverse dictionary is consulted, the method is called "one time inverse consultation" or IC1. If n inverse dictionaries are consulted, the method is called "n time inverse consultation". As there is no significant difference in performance, we simply implemented IC1. Assuming that each element (x) of these two sets (SA, E) has a weight that is determined by the number of times it appears in the set that belongs (X), this weight is denoted as $\delta(X, x)$. In the same way, the number of common elements between SA and E is denoted as follows:

$$\delta(E, SA) = \sum_{x \in SA} \delta(E, x) \quad (1)$$

IC asks for more than one pivot word between source word s and translation candidate t. In our example:

$$\delta(D_{a\rightarrow b}(s), D_{c\rightarrow b}(t)) > 1 \quad (2)$$

In general, this condition guarantees that pivot words belong to the same sense of the source word (e.g. *iturri→tap→grifo, iturri→faucet→grifo*). Consequently, source word and target word also belong to the same sense.

Conceptually, the IC method is based on the confluence of two evidences. Let us take our dictionaries as examples. If two or more pivot words share a translation t in the $D_{es\rightarrow en}$ dictionary ($|tr(t_c, D_{es\rightarrow en}| > 1$) (e.g. *grifo→tap, grifo→faucet*) we could hypothesize that they are lexical variants belonging to a unique sense c. If an entry s includes those translations ($|tr(s_c, D_{eu\rightarrow en})| > 1$) (e.g. *iturri→tap, iturri→faucet*)) in the $D_{eu\rightarrow en}$ dictionary, we could also hypothesize the same. We can conclude that entry s and candidate t are mutual translations because the hypothesis that *"faucet"* and *"tap"* are lexical variants of the same sense c is contrasted against two evidences. This makes IC highly dependant on the number of lexical variants. Specifically, IC needs several lexical variants in the pivot language per each entry sense in both dictionaries. Assuming that wrong pairs cannot fulfill this requirement (see Formula 2) we can estimate the probabilities of the conditions for solving an ambiguous pair (s, t) where s and $t \in c$, as follows:

(a) $p(|tr(s_c, D_{a\rightarrow b})| > 1)$: Estimated by computing the average coverage of lexical variants in the pivot language for each entry in $D_{a\rightarrow b}$.

(b) $p(|tr(t_c, D_{c\rightarrow b})| > 1)$: Estimated by computing the average coverage of lexical variants in the pivot language for each entry in $D_{c\rightarrow b}$.

(c) $p(|tr(s_c, D_{a\rightarrow b}) \bigcap tr(t_c, D_{c\rightarrow b})| > 1)$: Convergence degree between translations of s and t in $D_{a\rightarrow b}$ and $D_{c\rightarrow b}$ corresponding to c.

So, in order to obtain a good performance with IC, the dictionaries used need to provide a high coverage of lexical variants per sense in the pivot language. If we assume that variants of a sense do not vary considerably between dictionaries, performance of IC in terms of recall would be estimated as follows:

$$R = p(|tr(s_c, D_{a\rightarrow b})| > 1) * p(|tr(t_c, D_{c\rightarrow b})| > 1) \quad (3)$$

We estimated the adequacy of the different dictionaries in the experimental setup according to estimations (a) and (b). Average coverage of lexical variants in the pivot language was calculated for both dictionaries. It was possible because lexical variants in the target language were grouped according to senses in both dictionaries. Only ambiguous entries were analyzed because they are the set of entries which IC must solve. In the $D_{eu\rightarrow en}$ dictionary more than 75% of senses have more than one lexical variant in the pivot language. So, $p(|tr(s_c, D_{eu\rightarrow en})| > 1) = 0.75$. In $D_{es\rightarrow en}$ this percentage (23%) is much lower. So, $p(|tr(t_c, D_{es\rightarrow en})| > 1) = 0.23$. Therefore, $D_{eu\rightarrow en}$ dictionary is more suited to the IC method than $D_{es\rightarrow en}$. As the conditions must be met in the maximum of both dictionaries, performance according to Formula 3 would be: $0.75 * 0.23 = 0.17$. This means that IC alone could solve about 17% of ambiguous entries.

5.2 Distributional Similarity

DS has been used successfully for extracting bilingual terminology from comparable corpora. The underlying idea is to identify as translation equivalents those words which show similar distributions or contexts across two corpora of different languages, assuming that this similarity is proportional to the semantic distance. In other words, establishing an equivalence between cross lingual semantic distance and translation probability. This technique can be used for pruning wrong translations produced in a pivot-based dictionary building process (Kaji et al., 2008; Gamallo and Pichel, 2010).

We used the traditional approach to compute DS (Fung, 1995; Rapp, 1999). Following the "bag-of-words" paradigm, the contexts of a word w

are represented by weighted collections of words. Those words are delimited by a window (± 5 words around w) and punctuation marks. The context words are weighted with regard to w according to the Log-likelihood ratio measure, and the context vector of w is formed. After representing word contexts in both languages, the algorithm computes for each source word the similarity between its context vector and all the context vectors corresponding to words in the target language by means of the cosine measure. To be able to compute the cross-lingual similarity, the context vectors are put in the same space by translating the vectors of the source words into the target language. This is done by using a seed bilingual dictionary. The problem is that we do not have that bilingual dictionary, since that is precisely the one we are trying to build. We propose that dictionaries extracted from our noisy dictionary ($D_{eu \rightarrow en \rightarrow es}$) be used:

- Including the unambiguous entries only
- Including unambiguous entries and selecting the most frequent candidates according to the target language corpus for ambiguous entries
- The dictionary produced by the IC1 method

The second method performed better in the tests we carried out. So, that is the method implemented for the experiments in the next section.

DS calls for several conditions in order to perform well. For solving an ambiguous translation t of a source word s, both context representations must be accurate. The higher their frequency in the comparable corpus, the richer their context representation will be. In addition to context representation, the translation quality of contexts is also a critical factor for the performance of DS. Factors can be formulated as follows if we assume big and highly comparable corpora:

(a) Precision of context representation: this can be estimated by computing the frequency of the words

(b) Precision of translation process: this can be estimated by computing the quality of the seed dictionary

6 Results

In order to evaluate the performance of each pruning method, the quality of the translations was measured according to the average precision and recall of translations per entry with respect to the reference dictionary. As we were not interested in dealing with missing translations, the reference for calculating recall was drawn up with respect to the intersection between the merged dictionary ($D_{eu \rightarrow en \rightarrow es}$) and the reference dictionary ($D_{eu \rightarrow es}$). F-score is the metric that combines both precision and recall.

We also introduced the frequency of use of both entry and pair as an aspect to take into account in the analysis of the results. It is better to deal effectively with frequent words and frequent translations than rare ones. Frequency of use of Basque words and frequency of source-target translation equivalent pairs were extracted respectively from the open domain monolingual corpus and the parallel corpus described in the previous section. Corpora were lemmatized and POS tagged in both cases in order to extract the frequency information of the lemmas.

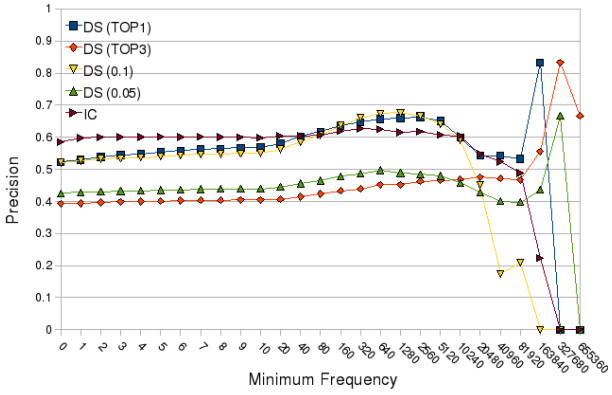


Figure 2: Precision results according to the minimum frequency of entries.

6.1 Inverse Consultation

Results show that IC precision is about 0.6 (See Figure 2). This means that many wrong pairs fulfill IC conditions. After analyzing the wrong pairs by hand, we observed that some of them corresponded to correct pairs not included in the reference dictionary. They are not included in

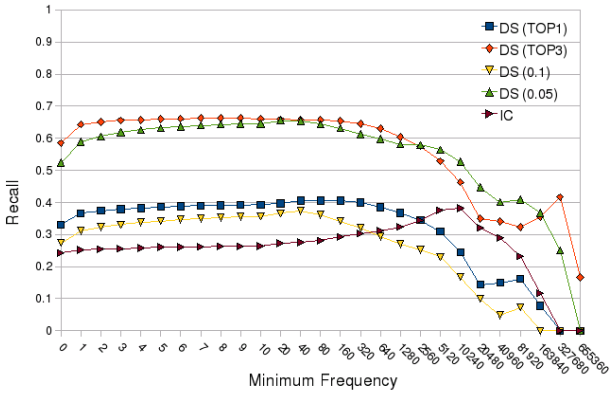


Figure 3: Recall results according to the minimum frequency of entries.

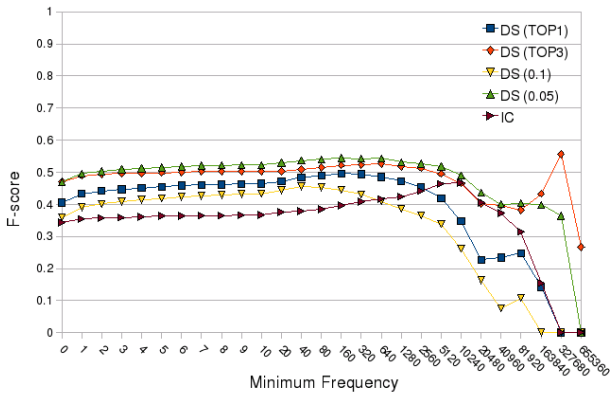


Figure 4: F-score results according to the minimum frequency of entries.

the reference because not all synonyms -or lexical variants- are included in it, only the most common ones. This is an inherent problem in automatic evaluation, and affects all the experiments presented throughout section 6 equally. Other wrong pairs comprise translation equivalents which have the same stem but different gramatical categories (e.g., 'aldakuntza' (noun) *(change, shift)* $\rightarrow$ *'cambiar'* (verb) *(to change, to shift)*). These wrong cases could be filtered if POS information would be available in the source dictionaries.

Precision is slightly better when dealing with frequent words, a maximum of 0.62 is reached when minimum frequency is between 150 and 2,000. Precision starts to decline significantly when dealing

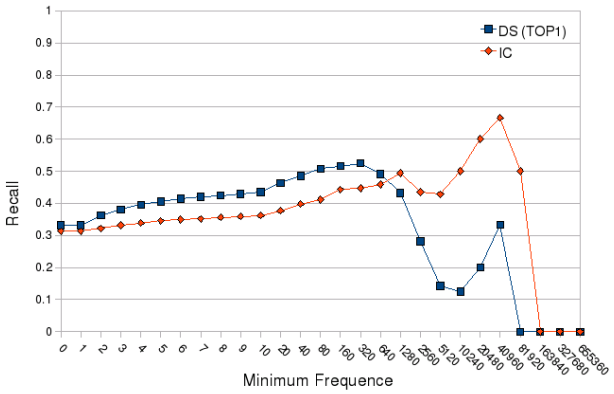


Figure 5: Recall results according to the minimum frequency of translation pairs.

with those entries over a minimum frequency of 10,000. However, only very few entries (234) reach that minimum frequency.

Recall is about 0.2 (See Figure 3), close to the estimation computed in section 5.1. It presents a more marked variability according to the frequency of entries, improving the performance as the frecuency increases. This could be due to the fact that frequent entries tend to have more translation variants (See Table 3). The fact that there are too many candidates to solve would explain why the recall starts to decline when dealing with very frequent entries.

Global performance according to F-score reflects the variability depending on frequency (See Figure 4).

Recall according to frequency of pairs provides information about whether IC selects rare translations or the most probable ones (See Figure 5). It must be noted that this recall is calculated with respect to the translation pairs of the merged dictionary $D_{eu \rightarrow en \rightarrow es}$ which appear in the parallel corpus (see section 4.1). Results (See Figure 5) show that IC deals much better with frequent translation pairs. However, recall for pairs whose frequency is higher than 100 only reaches 0.5. Even if the maximum recall is achieved for pairs whose frequency is above 40,000, it is not significant because they suppose a minimum number (3 pairs). In short, we can conclude that IC often does not find the most probable translation

(e.g. 'usain'→*'olor' (smell)*, 'zulo'→*'agujero' (hole)*,...).

6.2 Distributional Similarity

DS provides an idea of semantic distance. However, in order to determine whether a candidate is a correct translation, a minimum threshold must be established. It is very difficult to establish a threshold manually because its performance depends on the characteristics of the corpora and the seed dictionaries. The threshold can be applied at a global level, by establishing a numeric threshold for all candidates, or at local level by selecting certain top ranked candidates for each entry. The dictionary created by IC or unambiguous pairs can be used as a reference for tuning the threshold in a robust way with respect to the evaluation score such as F-score. In our experiments, thresholds estimated against the dictionary created by IC are very close to those calculated with respect to the whole reference dictionary (see Figure 6).

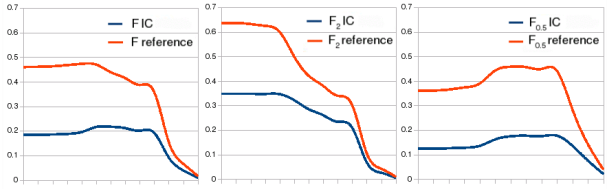


Figure 6: Threshold parameter tuning comparison for different F_n scores. Tuning against dictionary created by IC vs. Reference dictionary.

There is not much variation in performance between local and global thresholds. Precision increases from 0.4 to 0.5 depending on the strictness level of the threshold (See Figure 2), the stricter the better. In all cases, precision is slightly better when dealing with frequent words (frequency $>$ 20). This improvement is more marked with the strictest thresholds (TOP_1, 0.1). However, if global thresholds are used, performance starts to decline significantly when dealing with words whose frequency is above 1,000. So, it seems that local thresholds (TOP_3) perform more consistently with respect to the high frequencies of entries.

Recall (See Figure 3) goes from 0.5 to 0.7 depending on the strictness level of the threshold. It starts declining when frequency is above 50 depending on the type of threshold. In this case, global thresholds seem to perform better because the most frequent entries are handled better. These entries tend to have many translations. Therefore thresholds based on top ranks are too rigid.

There is no significant difference between global and local thresholds in terms of F-Score (See Figure 4). Each threshold type is more stable in precision or recall. So the F-Score is similar for both. Variability of F-Score according to frequency is lower than in precision and recall. As performance peaks on both measures at different points of frequency, the variability is mitigated when measures are combined by F-Score.

We have plotted the recall according to the frequency of pairs calculated from a parallel corpus in order to analyze the performance of DS when dealing with frequent translation pairs (See Figure 5). The performance decreases when dealing with pairs whose frequency is higher than 100. This means that DSs performance is worse when dealing with the most common translation pairs. So it is clear that it is very difficult to represent the contexts of very frequent words correctly.

The results show that DS rankings are worse when dealing with some words above a certain frequency threshold (e.g. 'on' *'good'*, 'berriz' *'again'*, 'buru' *'head'*, 'orain' *'now'*...). Although context representation of frequent words is based on many evidences, high polysemy level related to high frequency leads to a poorer representation. Alternatively we found that some of those frequent words are not very polysemous. Those words do not have strong collocates, that is, they tend to appear freely in contexts, which also leads to poor representation. This low quality representation hampers an accurate computation of semantic distance.

6.3 Comparison between IC and DS

As for average precision, IC provides better results than DS if all entries are taken into account. However, DS tips the scales in its favor if only entries with frequencies above 50 are considered and strict thresholds are used (TOP_1, 0.1).

DS clearly outperforms IC in terms of average recall of translations. Even if strict thresholds are used, DS outperforms IC for all entries whose

frequency is lower than 640.

If average precision and recall are evaluated together by means of F-score, DS outperforms IC (Figure 4). Only when dealing with very frequent entries (frequency $> 8,000$) is ICs performance close to DSs, but these entries make up a very small group (234 entries).

In order to compare the recall with respect to the frequency of translation pairs under the same conditions, we have to select a threshold that provides a similar precision to IC. TOP_1 is the most similar one (see figure 2). As Figure 5 shows, again DS is better than IC. Even if IC's recall clearly surpasses DS's when dealing with frequent translation pairs (frequency $> 2,560$), it only represents a minimal number of pairs (39).

6.4 Combining IC and DS according to different scenarios

In order to see how the methods can complement each other, we calculated the performance for solving ambiguous entries obtained by combining the results of both methods using various alternatives:

- Union: $IC \cup DS$: Pairs obtained by both methods are merged. Duplicated pairs are cleaned.
- Lineal combination (Lcomb): $IC * k + DS * (1 - k)$. Each method provides a value representing the translation score. For IC that value is the number of pivot words (see Formula 1), and the context similarity score in the case of DS. Those values are linearly combined and applied over the noised dictionary.

As mentioned in the first section, one of the goals of the paper was to analyze which method and which combination was best depending on the use case. We have selected some measures which are a good indicator of good performance for different use cases:

- $AvgF$: Average F-score per entry.
- $wAvgF$: Average F-score per entry weighted by the frequency of the entry. Higher frequency increases the weight.
- $AvgF_2$: Average F-score per entry where recall is weighted higher.
- $AvgF_{0.5}$: Average F-score per entry where precision is weighted higher.

For the use cases presented in section 1, some measures will provide richer information than others. On the one hand, if we aim to build small, accurate dictionaries, $AvgF_{0.5}$ would be a better indicator since it attaches more importance to high precision. In addition, if we want the dictionaries to cover the most common entries (e.g., in a basic dictionary for language learners) it is also interesting to look at $wAvgF$ values because greater value is given to finding translations for the most frequent words. On the other hand, if our objective is to build big dictionaries with a high recall, it would be better to look at $AvgF_2$ measure which attaches importance to recall.

Method	$AvgF$	$wAvgF$	$AvgF_2$	$AvgF_{0.5}$
IC	0.34	0.27	0.27	0.46
DS	0.47	0.44	0.64	0.46
Union	**0.52**	**0.49**	0.65	0.49
Lcomb	**0.52**	**0.49**	**0.67**	**0.52**

Table 3: Performance results of methods for ambiguous entries according to different measures.

Table 3 shows the results for the different combinations. The parameters of all methods are optimized for each metric (as explained in section 6.2, see figure 6). In all cases, the combinations surpass the results of both methods separately. There is a reasonable improvement over DS (10.6% for $AvgF$), and an even more startling one over IC (52.9% for $AvgF$). IC only gets anywhere near the other methods when precision is given priority ($AvgF_{0.5}$). There is no significant difference in terms of performance between the two combinations, although Lcomb is slightly better. $wAvgF$ measure is stricter than the others since it takes frequency of entries into account. This is emphasised more in the case of IC where results decrease notably compared with $AvgF$.

7 Conclusions

This paper has analyzed IC and DS, for the task of pruning wrong translations from bilingual dictionaries built by means of pivot techniques. After analyzing their strong and weak points we have showed that IC requires high ambiguity level dictionaries with several lexical variants per entry sense. With an average ambiguity close to 2 translation candidates DS obtains better results. IC is a high precision method, but contrary to our expectations, it seems that it is not much more precise than DS. In addition, DS offers much better recall of translations and entries. As a result, DS performs the best if both precision and recall are taken into account by F-score.

Both methods prune most probable translations for a significant number of frequent entries. DS encounters a problem when dealing with very frequent words due to the difficulty in representing their context. The main reason behind this is the high polysemy level of those words.

Our initial beliefs were that the translations found by each method would diverge to a certain extent. The results obtained when combining the two methods show that although the performance does not increase as much as expected (10.6% improvement over DS), there is in fact some divergence. As for the different use cases proposed, combinations offer the best performance in all cases. IC is indeed the poorer method, although it presents competitive results when precision is given priority.

Future experiments include contrasting these results with other dictionaries and language pairs.

8 Aknowledgments

This work has been partially founded by the Industry Department of the Basque Government under grants IE09-262 (Berbatek project) and SA-2010/00245 (Pibolex+ project).

References

Francis Bond and Kentaro Ogura. 2007. Combining linguistic resources to create a machine-tractable Japanese-Malay dictionary. Language Resources and Evaluation, 42(2):127–136.

Francis Bond, Ruhaida Binti Sulong, Takefumi Yamazaki, and Kentaro Ogura. 2001. Design and construction of a machine-tractable Japanese-Malay dictionary. Proceedings of ASIALEX, SEOUL, 2001(2001):200–205.

Maike Erdmann, Kotaro Nakayama, Takahiro Hara, and Shojiro Nishio. 2008. An approach for extracting bilingual terminology from wikipedia. In Proceedings of the 13th international conference on Database systems for advanced applications, DASFAA'08, pages 380–392, Berlin, Heidelberg. Springer-Verlag. ACM ID: 1802552.

Pascale Fung. 1995. Compiling bilingual lexicon entries from a non-parallel English-Chinese corpus. In David Yarovsky and Kenneth Church, editors, Proceedings of the Third Workshop on Very Large Corpora, pages 173–183, Somerset, New Jersey. Association for Computational Linguistics.

Pablo Gamallo and José Pichel. 2010. Automatic generation of bilingual dictionaries using intermediary languages and comparable corpora. In Alexander Gelbukh, editor, Computational Linguistics and Intelligent Text Processing, 11th International Conference, CICLing 2010. Proceedings, volume 6008 of Lecture Notes in Computer Science, pages 473–483. Springer.

Varga István and Yokoyama Shoichi. 2009. Bilingual dictionary generation for low-resourced language pairs. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2, EMNLP '09, pages 862–870, Stroudsburg, PA, USA. Association for Computational Linguistics. ACM ID: 1699625.

Hiroyuki Kaji, Shin'ichi Tamamura, and Dashtseren Erdenebat. 2008. Automatic construction of a Japanese-Chinese dictionary via English. In Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), Marrakech, Morocco. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Mausam, Stephen Soderland, Oren Etzioni, Daniel S Weld, Michael Skinner, and Jeff Bilmes. 2009. Compiling a massive, multilingual dictionary via probabilistic inference. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1, ACL '09, page 262270, Stroudsburg, PA, USA. Association for Computational Linguistics. ACM ID: 1687917.

Kyonghee Paik, Satoshi Shirai, and Hiromi Nakaiwa. 2004. Automatic construction of a transfer dictionary considering directionality. In Proceedings of the Workshop on Multilingual Linguistic Ressources, MLR '04, pages 31–38, Stroudsburg, PA, USA.

Association for Computational Linguistics. ACM ID: 1706243.

R. Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In Proceedings of the 37th annual meeting of the Association for Computational Linguistics, pages 519–526, College Park, USA. ACL.

Daphna Shezaf and Ari Rappoport. 2010. Bilingual lexicon generation using non-aligned signatures. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, page 98107, Stroudsburg, PA, USA. Association for Computational Linguistics. ACM ID: 1858692.

S. Shirai and K. Yamamoto. 2001. Linking english words in two bilingual dictionaries to generate another language pair dictionary. In Proceedings of ICCPOL, pages 174–179.

J. Sjöbergh. 2005. Creating a free digital Japanese-Swedish lexicon. In Proceedings of PACLING 2005.

Kumiko Tanaka and Kyoji Umemura. 1994. Construction of a bilingual dictionary intermediated by a third language. In Proceedings of the 16th International Conference on Computational Linguistics (COLING'94), pages 297–303.

Takashi Tsunakawa, Naoaki Okazaki, and Jun'ichi Tsujii. 2008. Building bilingual lexicons using lexical translation probabilities via pivot languages. Proceedings of the Sixth International Language Resources and Evaluation (LREC'08).

Soft Dependency Constraints for Reordering in Hierarchical Phrase-Based Translation

Yang Gao, Philipp Koehn and Alexandra Birch
School of Informatics
University of Edinburgh
Edinburgh, UK, EH8 9AB
yanggao1119@gmail.com, pkoehn@inf.ed.ac.uk, a.birch@ed.ac.uk

Abstract

Long-distance reordering remains one of the biggest challenges facing machine translation. We derive soft constraints from the source dependency parsing to directly address the reordering problem for the hierarchical phrase-based model. Our approach significantly improves Chinese–English machine translation on a large-scale task by 0.84 BLEU points on average. Moreover, when we switch the tuning function from BLEU to the LRscore which promotes reordering, we observe total improvements of 1.21 BLEU, 1.30 LRscore and 3.36 TER over the baseline. On average our approach improves reordering precision and recall by 6.9 and 0.3 absolute points, respectively, and is found to be especially effective for long-distance reodering.

1 Introduction

Reordering, especially movement over longer distances, continues to be a hard problem in statistical machine translation. It motivates much of the recent work on tree-based translation models, such as the hierarchical phrase-based model (Chiang, 2007) which extends the phrase-based model (Koehn et al., 2003) by allowing the so-called *hierarchical phrases* containing subphrases.

The hierarchical phrase-based model captures the recursiveness of language without relying on syntactic annotation, and promises better reordering than the phrase-based model. However, Birch et al. (2009) find that although the hierarchical phrase-based model outperforms the phrase-based model in terms of medium-range reordering, it does equally poorly in long-distance reordering due to constraints to guarantee efficiency.

Syntax-based models that use phrase structure constituent labels as non-terminals in their transfer rules, exemplified by that of Galley et al. (2004), produce smarter and syntactically motivated reordering. However, when working with off-the-shelf tools for parsing and alignment, this approach may impose harsh limits on rule extraction and requires serious efforts of optimization (Wang et al., 2010).

An alternative approach is to augment the general hierarchical phrase-based model with soft syntactic constraints. Here, we derive three word-based, complementary constraints from the source dependency parsing, including:

- A dependency orientation feature, trained with maximum entropy on the word-aligned parallel data, which directly models the head-dependent orientation for source words;
- An integer-valued cohesion penalty that complements the dependency orientation feature, and fires when a word is not translated with its head. It measures derivation well-formedness and is used to indirectly help reordering;
- An auxiliary unaligned penalty feature that mitigates search error given the other two features.

We achieve significant improvements in terms of the overall translation quality and reordering behavior. To our knowledge we are the first to use the source dependency parsing to target the reordering problem for hierarchical phrase-based MT.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 857–868,
Edinburgh, Scotland, UK, July 27–31, 2011.

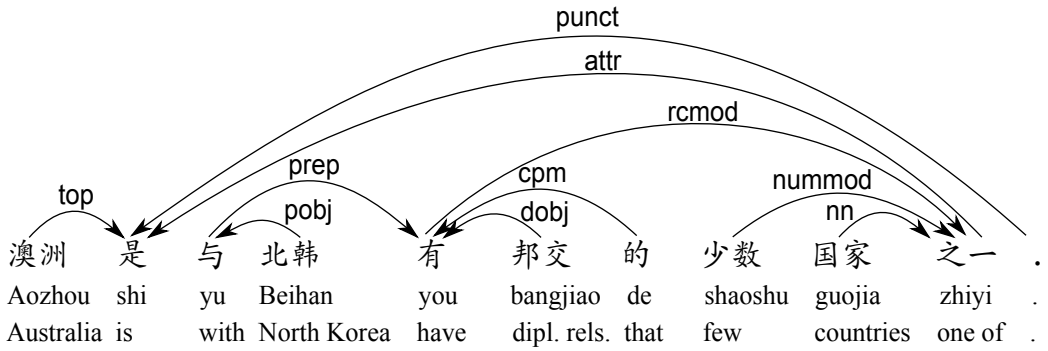


Figure 1: Example dependency parsing generated by the Stanford Parser. The Chinese source sentence and its English translation come from (Chiang, 2007).

2 Three Soft Dependency Constraints

Our features are based on the source dependency parsing, as shown in Figure 1. The basic unit of dependency parsing is a triple consisting of the dependent word, the head word and the dependency relation that connects them. For example, in Figure 1, an arrow labelled prep goes from the word *yu* (English *with*) to the word *you* (English *have*), showing that *yu* is a prepositional modifier of *you*.

We use the Stanford Parser[1] to generate dependency parsing, which automatically extracts dependency relations from phrase structure parsing (de Marneffe et al., 2006).

2.1 Dependency Orientation

Based on the assumption that constituents generally move as a whole (Quirk et al., 2005), we decompose the sentence reordering probability into the reordering probability for each aligned source word with respect to its head, excluding the root word at the top of the dependency hierarchy which does not have a head word. Similarly, Hayashi et al. (2010) also take a word-based reordering approach for HPBMT, but they model *all possible* pairwise orientation from the source side as a general linear ordering problem (Tromble and Eisner, 2009).

To be more specific, we have a maximum entropy orientation classifier that predicts the probability of a source word being translated in a monotone or reversed manner with respect to its head. For example, given the alignment in Figure 2(a), with the alignment points (i_{dep}, j_{dep}) for the source dependent word and (i_{head}, j_{head}) for the source head word, we define two orientation classes as:

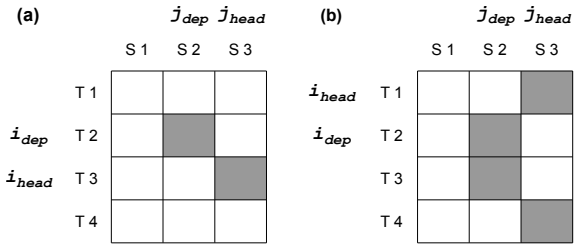


Figure 2: Word alignments to illustrate orientation classification. In (a), monotone (M); in (b), reversed (R).

$$c = \begin{cases} R & \text{if } (j_{dep} - j_{head})(i_{dep} - i_{head}) < 0 \\ M & \text{otherwise} \end{cases} \quad (1)$$

When a source head or dependent word is aligned to multiple target words, as shown in Figure 2(b), we always take the first target word for orientation classification.

The orientation classifier is trained on the large word-aligned parallel corpus. Various features can potentially be used, based on the source and target context as well as syntactic and semantic analysis. The orientation probability is evaluated in the following log-linear equation, where f is the source context, d is the source dependency parsing, e^* is the target context produced so far, a^* is the alignment produced so far and c is the orientation class:

[1] http://nlp.stanford.edu/software/lex-parser.shtml

Word	p(M)	p(R)
Aozhou	0.81	0.19
shi	NA	NA
yu	0.45	0.55
Beihan	0.88	0.12
you	0.12	0.88
bangjiao	0.83	0.17
de	0.58	0.42
shaoshu	0.30	0.70
guojia	0.19	0.81
zhiyi	0.85	0.15
.	1.00	0.00

Table 1: The dependency orientation probabilities for words of the Figure 1 sentence, in both monotone and reversed cases.

$$p(c|f,d,e^*,a^*) = \frac{exp(\sum_{n=1}^{N}\lambda_n h_n(f,d,e^*,a^*,c))}{\sum_{c'\in\{M,R\}} exp(\sum_{n=1}^{N}\lambda_n h_n(f,d,e^*,a^*,c'))} \quad (2)$$

Currently, we only use two kinds of features: (1) the concatenation of the source dependent word with the dependency relation and (2) the concatenation of the source head word with the dependency relation. So for the word *yu* (English *with*) in Figure 1, we extract these features for orientation classification: prep_DEP_*yu* and prep_HEAD_*you*.

We define the dependency orientation feature score for a translation hypothesis as the sum of the log orientation probabilities for each source word. This score is used as one feature in the log-linear formulation of the hierarchical phrase-based model.

Table 1 shows the dependency orientation probabilities for all words in the Figure 1 sentence. Most interestingly, the orientation probabilities for *you* (English *have*) strongly support global reordering of *one of the few countries* with the relative clause *that have diplomatic relations with North Korea*. We find that it is a general trend for long-distance reordering to gain stronger support, since it is often correlated with prominent reordering patterns (such as relative clause and preposition) as well as lexical evidences (such as *"... zhiyi"* (English *"one of ..."*)) for which the reversed orientation takes up the majority of the training cases.

Consider the following rules (both terminals and nonterminals are coindexed):

$$X \rightarrow (\underline{yu_1\ Beihan_2}\ you_3\ bangjiao_4,\ have_3\ dipl._4\ rels._4\ \underline{with_1\ North_2\ Korea_2}) \quad (3)$$

$$X \rightarrow (\underline{yu_1\ Beihan_2}\ you_3\ bangjiao_4,\ \underline{with_1\ North_2\ Korea_2}\ have_3\ dipl._4\ rels._4) \quad (4)$$

According to Table 1, the hypothesis that applies Rule 3 receives a probability of 0.55 for *yu* getting reversed with its head *you*, as well as 0.88 and 0.83 for translating *Beihan* and *bangjiao* in a monotone manner with respect to their heads. Rule 4 is associated with probabilities 0.45, 0.88 and 0.83 for monotone translation of *yu*, *Beihan* and *bangjiao*. Thus our dependency orientation feature is able to trace the difference in ordering the PP *with North Korea* (as underlined) and the VP *have dipl. rels.* down to the orientation of the preposition *yu* (English *with*) with respect to its head *you* (English *have*), and promote Rule 3 which has the right word order.

The word *you* (English *have*) cannot be scored in Rules 3 or 4, since its head word *zhiyi* (English *one of*) is not covered. In this case, we say that the word *you* is unresolved. We carry an unresolved word along in the derivation process until we reach a terminator hypothesis which translates the head word. Then the resulting dependency orientation score is added to the terminator hypothesis. This means that the dependency orientation feature is "stateless", i.e., hypotheses that cover the same source span with the same orientation information will receive the same feature score, regardless of the derivation history. Therefore, Derivation 5 in the following will have the same dependency orientation score as Derivation (Rule) 3, and Derivation 6 will score the same as Derivation (Rule) 4.

5.1 $X \rightarrow (yu_1, with_1)$

5.2 $X \rightarrow (Beihan_1, North_1\ Korea_1)$

5.3 $X \rightarrow (X_1\ X_2, X_1\ X_2)$

5.4 $X \rightarrow (X_1\ you_2\ bangjiao_3,\ have_2\ dipl._3\ rels._3\ X_1)$ (5)

6.1 $X \rightarrow$ (*Beihan*$_1$ *you*$_2$, *North*$_1$ *Korea*$_1$ *has*$_2$)

6.2 $X \rightarrow$ (X_1 *bangjiao*$_2$, X_1 *dipl.*$_2$ *rels.*$_2$)

6.3 $X \rightarrow$ (*yu*$_1$ X_2, *with*$_1$ X_2)

(6)

2.2 Cohesion Penalty

When the dependency orientation for a word is temporarily unavailable ("unresolved"), a cohesion penalty fires. Cohesion penalty counts the total occurrences of unresolved words for a translation hypothesis, which involve newly encountered unresolved words as well as old unresolved words carried on from the derivation history. Therefore, the cohesion penalty is "stateful", i.e., an unresolved word is repeatedly penalized until it gets resolved. Under this definition, the most cohesive derivation translates the entire sentence with one rule, where every word is locally resolved. The least cohesive derivation translates each word individually and glues word translations together. Consulting Figure 1, the cohesion penalty in Derivation 5 is 4, since the word *yu* (English *with*) is unresolved twice (in 5.1 and 5.3), and both *Beihan* (English *North Korea*) and *you* (English *have*) are unresolved once (in 5.2 and 5.4, respectively); the cohesion penalty in Derivation 6 is 5: 2 from *Beihan* (English *North Korea*) (in 6.1 and 6.2) and 3 from *you* (English *have*). As a result, Derivation 5 gets promoted, which echoes with human intuition since Derivation 5 translates syntactic constituents. To sum up, our cohesion penalty provides an integer-valued measure of derivation well-formedness in the hierarchical phrase-based MT. Same as dependency orientation, the cohesion penalty is not applicable to the root word of the sentence.

We propose the cohesion penalty in order to further improve reordering, especially in long-distance cases, since a well-formed derivation at an earlier stage makes it more likely to explore hierarchical rules that perform more reliable reordering. In this respect, the cohesion penalty can be seen as an aid to the glue rule penalty and as an alternative to constituency-based constraints.

Specifically, the glue rule penalty (Chiang, 2007) promotes hierarchical rules. Hierarchical rules whose lexical evidence helps resolve words locally will also be favored by our cohesion penalty feature. However, ignorant of the syntactic structure, the glue rule penalty may penalize a reasonably cohesive derivation such as Derivation 5 and at the same time promote a less cohesive hierarchical translation, such as Derivation 6.

Compared with constituency constraints based on the phrase structure, our cohesion penalty derived from the binary dependency parsing has two different characteristics.

First, our cohesion penalty is by nature more tolerant to some meaningful noncontituent translations. For example, constituency constraints in (Chiang, 2005; Marton and Resnik, 2008; Chiang et al., 2009) would penalize Rule 7 below which is useful for German–English translation (Koehn et al., 2003), and Rule 8 which can be applied to the Figure 1 sentence. Fuzzy constituency constraints can solve this problem with a combination of product categories and slash categories (Chiang, 2010). Yet our cohesion penalty by nature admits these translations as cohesive (with no extra cost from *es* and *Aozhou* since both are locally resolved). Admittedly, our current implementation of the cohesion penalty is blind to some other meaningful nonconstituent collocations, such as neighbouring siblings of a common uncovered head (regulated as the "floating structure" in (Shen et al., 2008)). A concrete example is Rule 9 which is useful for the Figure 1 sentence. To address this problem, another feature can be defined in the same manner to capture how each head word is translated with its children.

$$X \rightarrow (es_1\ gibt_2,\ there_1\ is_2) \quad (7)$$

$$X \rightarrow (Aozhou_1\ shi_2,\ Australia_1\ is_2) \quad (8)$$

$$X \rightarrow (shaoshu_1\ guojia_2,\ few_1\ countries_2) \quad (9)$$

Second, our cohesion penalty can be by nature more discriminative. Compared with the constituency constraints, the cohesion penalty is integer-valued, and can be made sensitive to the depth of each word in the dependency hierarchy (see Section 2.4). Inspired by (Marton and Resnik, 2008; Chiang et al., 2009), the cohesion penalty could also be made sensitive to the dependency relation of each word. However, this drastically increases the number of features and requires a tuning algorithm which scales better to high-dimensional model spaces, such as MIRA (Watanabe et al., 2007; Chiang et al., 2008).

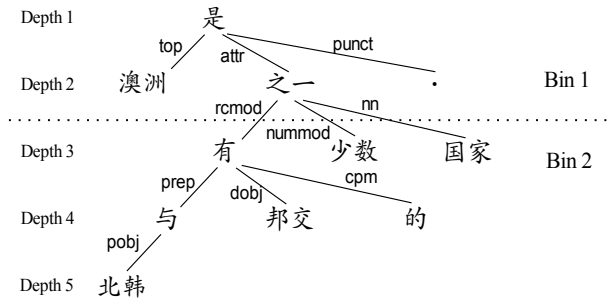


Figure 3: Using 2 bins for the dependency parse tree of the Figure 1 sentence.

2.3 Unaligned Penalty

The dependency orientation and cohesion penalty cannot be applied to unaligned source words. This may lead to search error, such as dropping (i.e., unaligning) key content words that are important for lexical translation and reordering. The problem is mitigated by an unaligned penalty applicable to all words in the dependency hierarchy.

2.4 Grouping Words into Bins

Having defined dependency orientation, cohesion penalty and unaligned penalty, we section the source dependency tree *uniformly by depth*, group words at different depths into bins and only add the feature scores of a word into its respective bin. In this way one feature is split into several sub-features and each can be trained discriminatively by MERT.

There are two motivations for binning. The primary motivation is to distinguish long-distance reordering which is still problematic for the hiero-style model, since local reorderings generally operate at low levels of the tree while high tree levels tend to take more care of long-distance reordering. Parsing accuracy is another concern, yet its impact on feature performance is intricate and our MaxEnt-trained dependency orientation feature also buffers against odd parsing. Using bins, we simply let the tuning process decide how much to trust feature scores coming from different levels of parsing.

We experiment with 1, 2 and 3 bins. An example of binning for the Figure 1 sentence can be found in Figure 3. With 2 bins (hereafter "bin-2"), words at Depth 1 and 2 are grouped into Bin 1, and words at Depth 3, 4, 5 are grouped into Bin 2. As a simple approach, binning does not take into account how the tree levels spread out.

3 Experiments

3.1 General Settings

We used a parallel training corpus with 2.1 million Chinese–English sentence pairs, aligned by GIZA++. The Chinese side was parsed by the Stanford Parser. Then we extracted 33.8 million examples from the parsed Chinese side to discriminatively train 1.1 million features (using the MegaM software[2]) for dependency orientation classification.

We trained three 5-gram language models with modified Kneser-Ney smoothing (Kneser and Ney, 1995): one on the English half of the parallel corpus, one on the Xinhua part of the Gigaword corpus, one on the AFP part, and interpolated them for best fit to the tuning set (Schwenk and Koehn, 2008).

We used NIST MT06 evaluation data (1664 lines) as our tuning set, and tested on NIST MT02 (878 lines), MT05 (1082 lines) and MT08 (1357 lines).

Our baseline system was the Moses implementation of the hierarchical phrase-based model with standard settings (Hoang et al., 2009). When only 1 bin was used, 3 additional features were added to the baseline, one each from the soft dependency constraints. When we used 2 or 3 bins, the additional feature counts doubled or tripled. We preserved terminal alignment alongside nonterminal alignment during the rule extraction and output word alignments together with translated strings. Since the features we currently define are based entirely on the source side, we used preprocessing to speed up decoding of our feature-augmented model. All experiments were tuned with MERT (Och, 2003).

3.2 Using BLEU as the Tuning Metric

As a standard practice, we first used BLEU (Papineni et al., 2002) as the objective function for tuning. Table 2 shows the results of the baseline model as well as our complete feature-augmented model with different bin numbers. With the "bin-2" setting, we get substantial improvement of up to 1.03 BLEU points (on MT02 data), and 0.84 BLEU points on average. Using more than one bin (i.e., differentiating tree depths) is generally beneficial, although the

[2]http://www.umiacs.umd.edu/∼hal/megam/index.html

Setting	BLEU / LRscore / TER			
	MT02	MT05	MT08	Average
baseline	34.01 / 41.85 / 68.93	32.23 / 40.50 / 68.15	28.09 / 37.17 / 66.82	31.44 / 39.84 / 67.97
bin-2	**35.04 / 43.07 / 65.58**	**33.18 / 41.62 / 65.59**	**28.63 / 38.12 / 65.36**	**32.28 / 40.94 / 65.51**
baseline-lr	34.23 / 42.06 / 68.08	32.28 / 40.61 / 67.61	27.99 / 37.27 / 66.98	31.50 / 39.98 / 67.56
bin-2-lr	**35.42 / 43.25 / 64.82**	**33.44 / 41.80 / 64.88**	**29.10 / 38.38 / 64.14**	**32.65 / 41.14 / 64.61**

Table 4: Results for the baseline model and the complete feature-augmented model with 2 bins ("bin-2"), using BLEU and LRscore ("-lr") as the tuning function. The BLEU scores of "bin-2" and "bin-2-lr" are significantly better than baseline ($p < 0.05$), computed by paired bootstrap resampling (Koehn, 2004).

Setting	BLEU			
	MT02	MT05	MT08	Average
baseline	34.01	32.23	28.09	31.44
bin-1	34.20	32.13	28.41	31.58(+.14)
bin-2	**35.04**	**33.18**	**28.63**	**32.28(+.84)**
bin-3	34.35	32.79	28.37	31.84(+.40)

Table 2: Results of the baseline model as well as our complete feature-augmented model with 1, 2 and 3 bins. BLEU is the tuning function.

Setting	BLEU			
	MT02	MT05	MT08	Average
baseline	34.01	32.23	28.09	31.44
dep	34.26	32.58	28.07	31.64(+.20)
dep+coP	34.47	32.81	28.61	31.96(+.52)
dep+coP+unP	35.04	33.18	28.63	32.28(+.84)

Table 3: Contributions of the three soft dependency constraints, with the "bin-2" setting

problem of overfitting sets in when we use 3 bins (with slightly higher tuning BLEU, not shown here).

We also studied the effect of adding features incrementally onto the baseline with the "bin-2" setting, as shown in Table 3. On average, all three features seem to have similar contributions.

3.3 Using LRscore as the Tuning Metric

Since our features are proposed to address the reordering problem and BLEU is not sensitive enough to reordering (especially in long-distance cases), we have also tried tuning with a metric that highlights reordering, i.e., the LRscore (Birch and Osborne, 2010). LRscore is a linear interpolation of a lexical metric and a reordering metric. We interpolated BLEU (as the lexical metric) with the Kendall's tau permutation distance (as the reordering metric). The Kendall's tau permutation distance measures the relative word order difference between the translation output and the reference(s) and is particularly sensitive to long-distance reordering. Testing results in terms of BLEU, LRscore and TER (Snover et al., 2006) are shown in Table 4. Tuned with the LRscore, our feature-augmented model achieves further average improvements (compare "bin-2" and "bin-2-lr") of 0.20 LRscore *as well as* 0.37 BLEU and 0.90 TER. Note that while the BLEU increase can largely be seen as a projection of the LRscore increase back into its lexical component, the consistent TER drop confirms that our improvement is not metric-specific[3]. Altogether the final improvement is 1.21 BLEU, 1.30 LRscore and 3.36 TER on average over the baseline.

However, an important question is how our features affect short, medium and long-distance reorderings. In the next section, we conduct quantitative analysis on reordering precision and recall, as well as qualitative analysis on translation examples.

4 Analysis

4.1 Precision and Recall of Reordering

The key to obtaining precision and recall for reordering is to investigate whether reorderings in the references are reproduced in the translations. We calculate precision as the number of reproduced reorderings divided by the total number of reorderings in the translation, and recall as the number of reproduced reorderings divided by the number of reorder-

[3]One of our reviewers points out that according to the inductive learning theory, it is counter-intuitive to improve on BLEU and TER if we optimize by the LRscore. Yet we do observe some other papers reporting increased TER or other metric scores when BLEU is used for tuning (Carpuat and Wu, 2007; Shen et al., 2008), suggesting that MT evaluation might be too complicated to be characterized just with inductive learning. Similar results based on extensive experiments can also be found in (Birch and Osborne, 2011).

Setting	MT02	MT05	MT08	Average
baseline	37.0	35.3	35.6	36.0
bin-2	42.7	40.8	38.7	40.7 (+4.7)
baseline-lr	37.3	35.0	34.2	35.5 (-0.5)
bin-2-lr	44.1	42.0	42.5	42.9 (+6.9)

Table 5: Overall precision for the test sets.

Setting	MT02	MT05	MT08	Average
baseline	37.5	36.2	33.2	35.6
bin-2	36.8	35.9	31.8	34.8 (-0.8)
baseline-lr	37.0	35.6	32.2	34.9 (-0.7)
bin-2-lr	37.7	36.7	33.2	35.9 (+0.3)

Table 6: Overall recall for the test sets.

ings in the reference. Then we average the precision and recall over all four reference translations.

Details of measuring reproduced reordering can be found in Birch et al. (2008). An important difference in this work is in handling many-to-one and one-to-many alignments, as we only retain the first word alignment for any source or target word which has multiple alignments. This is consistent with our treatment in dependency orientation classification, and results in more reorderings being extracted.

From Table 5 we can see that our features improve precision by an average of 4.7 absolute points when BLEU is used for tuning ("bin-2"). Switching from BLEU to the LRscore ("bin-2-lr"), we gain 2.2 points more and have a total improvement of 6.9 absolute points on average. This is a novel and important finding as we directly show that the quality of reordering has been improved.

From Table 6, we observe a small but consistent increase in recall with the "bin-2-lr" setting, averaging 0.3 absolute points. However, the drop of recall with the "bin-2" setting (by an average 0.8 points from the baseline) is unexpected. It seems that when applying our features alone, we are trading a small drop in recall for a large gain in precision.

In Figure 4 we break down the precision and recall statistics in MT08 by the reordering width on the source side. We find that our features consistently help precision over all word ranges, with more substantial improvement in the medium and long word ranges. When recall is concerned, our model does not help for short ranges of up to Width 4, but improves consistently for longer distance re-

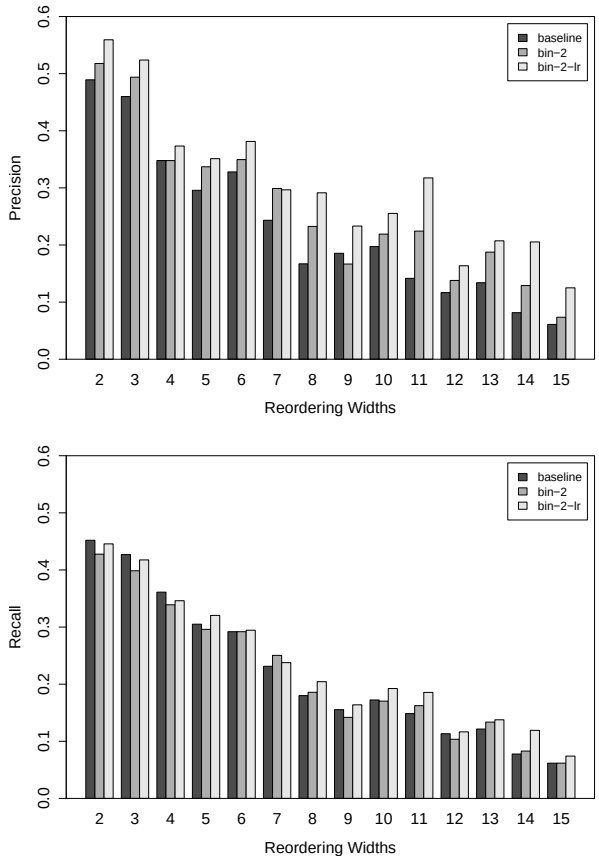
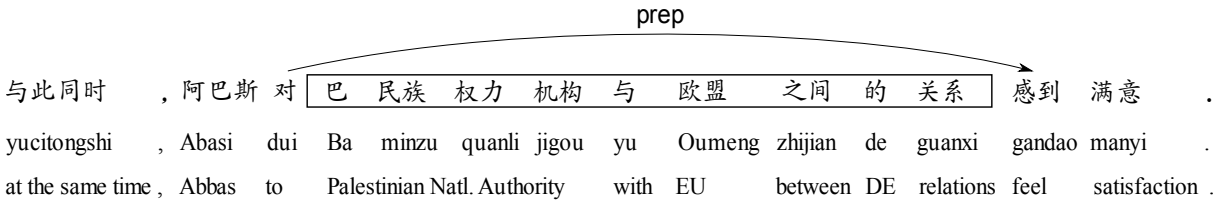


Figure 4: Precision and recall breakdown for the source-side reordering width 2-15 for the NIST MT08 dataset.

orderings. Once again, it seems that the feature-augmented model is able to benefit from tuning with a metric that is more sensitive to reordering, as the performance of "bin-2-lr" is the best in all reordering statistics.

4.2 Translation Examples

We observe a number of outputs with improved word order and more cohesive derivation, as the one in Figure 5. The baseline translation is fragmented and requires more glue rule applications. Specifically, it fails to translate the boxed area as a whole into "the relations between the palestinian national authority (pna) and the european union (eu)". The key dependency orientation that controls the global reordering is between the prepositional modifier *dui* (English *to*) and its head word, the verb *gandao* (English *feel*). The baseline system translates *dui* (English *to*) as "of the" and misorders the sentence. In contrast, the feature-augmented model "bin-2" cap-



baseline: [at the same time , abbas] [of the] [palestinian [national authority (pna)]] [and] [is satisfied with the [relations] [between [the european union (eu)]]] [.]

bin-2: [at the same time , abbas] [expressed satisfaction with [the relations between the [palestinian [national authority (pna) [and the european union (eu)]]]] .]

Figure 5: Example translations from the NIST MT08 set, output by the baseline model and "bin-2" model. The "-lr" version outputs are quite similar and not shown here. Translation outputs are in lower case.

tures the boxed area as a whole and uses Rule 10 to perform the right global reordering.

$$X \rightarrow (\underline{dui_1\ X_2}\ gandao_3\ manyi_4\ .5\ ,\ expressed_3\ satisfaction_4\ \underline{with_1\ X_2}\ .5\ ) \quad (10)$$

5 Related Work

In recent years, there has been a growing body of research on using dependency for statistical machine translation. Some directly encodes dependency in the translation model (Ding and Palmer, 2005; Quirk et al., 2005; Xiong et al., 2007; Shen et al., 2008; Mi and Liu, 2010), while others use dependency as a soft constraint (Cherry, 2008; Bach et al., 2009a,b; Chang et al., 2009). Among them, Shen et al. (2008) report that just filtering the phrase table by the so-called well-formed target dependency structure does not help, yet adding a target dependency language model improves performance significantly. Our intuitive interpretation is that the target dependency language model capitalizes on two characteristics of the dependency structure: it is based on words and it directly connects head and child. Therefore, the target dependency language model makes good use of the dependency representation as well as the target side training data.

We follow the second line of research, and derive three *word-based* soft constraints from the source dependency parsing. Note that although we reuse the word "cohesion" to name one of the constraints, our work is different from (Cherry, 2008; Bach et al., 2009a,b) which have successfully defined *another* cohesion constraint from the source dependency structure, with the aim of improving reordering in phrase-based MT.

To take a glance, Cherry (2008) and Bach et al. (2009b) define cohesion as translating a source dependency subtree contiguously into the target side without interruption (span or subtree overlapping), following Fox (2002). This span-based cohesion constraint has a different criterion from our word-based cohesion penalty and often leads to opposite conclusions. Bach et al. (2009a) also use cohesion to correlate with the lexicalized reordering model (Tillman, 2004; Koehn et al., 2005), whereas we define an orthogonal dependency orientation feature to explicitly model head-dependent reordering.

The fundamental difference, however, is rooted in the translation model. Their span-based cohesion constraint is implemented as an "interruption check" to encourage finishing a subtree before translating something else. This check is very effective for phrase-based decoding which searches over an entire space within the distortion limit in order to advance a hypothesis. In fact, it constrains reordering for the phrase-based model, as Cherry finds that the cohesion constraint is used "primarily to prevent distortion" and to provide "an intelligent estimate as to when source order must be respected" (Cherry, 2008). However, since the hierarchical phrase-based model *already* conducts principled reordering search with rules through the more constrained chart-decoding, ill-formed derivations exhibit themselves more often as nonconstituent translation than interrupted translation as defined in (Cherry, 2008; Bach et al., 2009a,b) (They do have a non-empty intersection, but neither subsumes the other). There-

fore, our cohesion penalty is better suited for the hierarchical phrase-based model.

To discourage nonconstituent translation, Chiang (2005) has proposed a constituency feature to examine whether a source rule span matches the source constituent as defined by phrase structure parsing. Finer-grained constituency constraints significantly improve hierarchical phrase-based MT when applied on the source side (Marton and Resnik, 2008; Chiang et al., 2009), or on the target side in a more tolerant fashion (Zollmann and Venugopal, 2006). Using both source and target syntax, but relaxing on rule extraction and substitution enables HPBMT to produce more well-formed and syntactically richer derivations (Chiang, 2010). Softening constituency matching with latent syntactic distributions proves to be helpful (Huang et al., 2010). Compared to constituency-based approaches, our cohesion penalty based on the dependency structure naturally supports constituent translations as well as some nonconstituent translations, if not all of them (as discussed in Section 2.2).

Our dependency orientation feature is similar to the order model within dependency treelet translation (Quirk et al., 2005). Yet instead of a head-relative position number for each modifier word, we simply predict the head-dependent orientation which is either monotone or reversed. Our coarser-grained approach is more robust from a machine learning perspective, yet still captures prominent and long-distance reordering patterns observed in Chinese–English (Wang et al., 2007), German–English (Collins et al., 2005), Japanese–English (Katz-Brown and Collins, 2008) and translation from English to a group of SOV languages (Xu et al., 2009). Not committed to specific language pairs, we learn orientation classification from the word-aligned parallel data through maximum entropy training as Zens and Ney (2006) and Chang et al. (2009) for phrase-based translation and Xiong et al. (2006) for the BTG model (Wu, 1996). While Chang et al. (2009) also make use of source dependency, their orientation classification concerns two subsequent phrase pairs in the left-to-right phrase-based decoding (as apposed to each dependent word and its head) and is therefore less linguistically-motivated.

6 Conclusion

We have derived three novel features from the source dependency structure for hierarchical phrase-based MT. They work as a whole to capitalize on two characteristics of the dependency representation: it is directly based on words and it directly connects head and child. The effectiveness of our approach has been demonstrated by a final average improvement of 1.21 BLEU, 1.30 LRscore and 3.36 TER. On average we improve reordering precision and recall by 6.9 and 0.3 absolute points, respectively, over the baseline. Moreover, our approach is found to be especially effective for long-distance reodering.

As mentioned in Section 2.2, the cohesion penalty can be extended to also account for how a head word is translated with its children so that we are not biased towards one form of cohesive nonconstituent translation. All our features can be made sensitive to the dependency relations or even words. This fine-grainedness is especially desirable when we want to reward words for being unaligned or unresolved, such as punctuations and function words in certain context. Word alignment quality is crucial for the performance of our features as well as the LRscore which uses word alignment to compute the permutation distance. As an alternative to GIZA++, we would like to experiment with syntactically informed aligners that better handle function words which often exhibit high alignment ambiguity due to low cross-lingual correspondence.

Finally, since our soft dependency constraints promote reordering without increasing model complexity, further gains can be achieved when combining our approach with orthogonal studies to improve the quantity and quality of hierarchical (reordering) rules, such as relaxing hierarchical rule extraction constraints (Setiawan and Resnik, 2010) and selectively lexicalizing rules with function words (Setiawan et al., 2009).

Acknowledgments

We would like to thank Miles Osborne, Adam Lopez, Barry Haddow, Hieu Hoang, Philip Williams and Michael Auli in the Edinburgh SMT group as well as Kevin Knight, David Chiang and Andrew Dai for inspiring discussions. We appreciate Pichuan Chang, Huihsin Tseng, Richard Zens, Matthew Snover and Nguyen Bach for helping us

understand their brilliant work. Many thanks to the anonymous reviewers for their insightful comments and suggestions. This work was supported in part by the EuroMatrixPlus project funded by the European Commission (7th Framework Programme) and in part under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022.

References

Bach, N., Gao, Q., and Vogel, S. (2009a). Source-side dependency tree reordering models with subtree movements and constraints. In *Proceedings of the Twelfth Machine Translation Summit (MTSummit-XII)*, Ottawa, Canada. International Association for Machine Translation.

Bach, N., Vogel, S., and Cherry, C. (2009b). Cohesive constraints in a beam search phrase-based decoder. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 1–4, Boulder, Colorado.

Birch, A., Blunsom, P., and Osborne, M. (2009). A quantitative analysis of reordering phenomena. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 197–205, Athens, Greece.

Birch, A. and Osborne, M. (2010). LRscore for evaluating lexical and reordering quality in MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 327–332, Uppsala, Sweden.

Birch, A. and Osborne, M. (2011). Reordering metrics for mt. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 1027–1035, Portland, Oregon, USA.

Birch, A., Osborne, M., and Koehn, P. (2008). Predicting success in machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 745–754, Honolulu, Hawaii.

Carpuat, M. and Wu, D. (2007). Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72, Prague, Czech Republic.

Chang, P.-C., Tseng, H., Jurafsky, D., and Manning, C. D. (2009). Discriminative reordering with Chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 51–59, Boulder, Colorado.

Cherry, C. (2008). Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 72–80, Columbus, Ohio.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Stroudsburg, PA, USA.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).

Chiang, D. (2010). Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden.

Chiang, D., Knight, K., and Wang, W. (2009). 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Boulder, Colorado.

Chiang, D., Marton, Y., and Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Honolulu, Hawaii.

Collins, M., Koehn, P., and Kucerova, I. (2005). Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan.

de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*.

Ding, Y. and Palmer, M. (2005). Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 541–548, Ann Arbor, Michigan.

Fox, H. (2002). Phrasal cohesion and statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 304–3111.

Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA.

Hayashi, K., Tsukada, H., Sudoh, K., Duh, K., and Yamamoto, S. (2010). Hierarchical phrase-based machine translation with word-based reordering model. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 439–446, Beijing, China.

Hoang, H., Koehn, P., and Lopez, A. (2009). A unified framework for phrase-based, hierarchical, and syntax-based statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 152–159, Tokyo, Japan.

Huang, Z., Cmejrek, M., and Zhou, B. (2010). Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 138–147, Cambridge, MA.

Katz-Brown, J. and Collins, M. (2008). Syntactic reordering in preprocessing for japanese-to-english translation: Mit system description for ntcir-7 patent translation task. In *Proceedings of NTCIR-7 Workshop Meeting*, Tokyo, Japan.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In Lin, D. and Wu, D., editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Koehn, P., Axelrod, A., Birch, A., Callison-burch, C., Osborne, M., and Talbot, D. (2005). Edinburgh system description for the 2005 iwslt speech translation evaluation. In *Proceedings of IWSLT2005*.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase based translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Marton, Y. and Resnik, P. (2008). Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-08: HLT*, pages 1003–1011, Columbus, Ohio.

Mi, H. and Liu, Q. (2010). Constituency to dependency translation with forests. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1433–1442, Uppsala, Sweden.

Och, F. J. (2003). Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL)*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Quirk, C., Menezes, A., and Cherry, C. (2005). Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, Ann Arbor, Michigan.

Schwenk, H. and Koehn, P. (2008). Large and diverse language models for statistical machine translation. In *Proceedings of International Joint Conference on Natural Language Processing*.

Setiawan, H., Kan, M. Y., Li, H., and Resnik, P. (2009). Topological ordering of function words in hierarchical phrase-based translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 324–332, Suntec, Singapore.

Setiawan, H. and Resnik, P. (2010). Generalizing hierarchical phrase-based translation using rules with adjacent nonterminals. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 349–352, Los Angeles, California.

Shen, L., Xu, J., and Weischedel, R. (2008). A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Tillman, C. (2004). A unigram orientation model for statistical machine translation. In *HLT-NAACL 2004: Short Papers*, pages 101–104, Boston, Massachusetts, USA.

Tromble, R. and Eisner, J. (2009). Learning linear ordering problems for better translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1007–1016, Singapore.

Wang, C., Collins, M., and Koehn, P. (2007). Chinese syntactic reordering for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and*

Computational Natural Language Learning (EMNLP-CoNLL), pages 737–745, Prague, Czech Republic.

Wang, W., May, J., Knight, K., and Marcu, D. (2010). Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36(2).

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic.

Wu, D. (1996). A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 152–158, Santa Cruz, California, USA.

Xiong, D., Liu, Q., and Lin, S. (2006). Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528, Sydney, Australia.

Xiong, D., Liu, Q., and Lin, S. (2007). A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 40–47, Prague, Czech Republic.

Xu, P., Kang, J., Ringgaard, M., and Och, F. (2009). Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 245–253, Boulder, Colorado.

Zens, R. and Ney, H. (2006). Discriminative reordering models for statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 55–63, New York City.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141, New York City.

Statistical Machine Translation with Local Language Models

Christof Monz
Informatics Institute, University of Amsterdam
P.O. Box 94323, 1090 GH Amsterdam, The Netherlands
c.monz@uva.nl

Abstract

Part-of-speech language modeling is commonly used as a component in statistical machine translation systems, but there is mixed evidence that its usage leads to significant improvements. We argue that its limited effectiveness is due to the lack of lexicalization. We introduce a new approach that builds a separate local language model for each word and part-of-speech pair. The resulting models lead to more context-sensitive probability distributions and we also exploit the fact that different local models are used to estimate the language model probability of each word during decoding. Our approach is evaluated for Arabic- and Chinese-to-English translation. We show that it leads to statistically significant improvements for multiple test sets and also across different genres, when compared against a competitive baseline and a system using a part-of-speech model.

1 Introduction

Language models are an important component of current statistical machine translation systems. They affect the selection of phrase translation candidates and reordering choices by estimating the probability that an application of a phrase translation is a fluent continuation of the current translation hypothesis. The size and domain of the language model can have a significant impact on translation quality. Brants et al. (2007) have shown that each doubling of the training data from the news domain (used to build the language model), leads to improvements of approximately 0.5 BLEU points. On the other hand, each doubling using general web data leads to improvements of approximately 0.15 BLEU points.

While large n-gram language models do lead to improved translation quality, they still lack any generalization beyond the surface forms (Schwenk, 2007). Consider example (1), which is a short sentence fragment from the MT09 Arabic-English test set, with the corresponding machine translation output (1.b), from a phrase-based statistical machine translation system, and reference translation (1.c).

(1) a. ... خلفية تصريحات صحافية مثيرة للجدل واتهامهم له ...

b. ... the background of press statements of controversial and accused him ...

c. ... the background of controversial press statements and accused him ...

Clearly, the adjective "controversial" should precede the nouns "press statement", but since the AFP and Xinhua portions of the Gigaword corpus, used to build the language model for the translation system, do not contain this surface n-gram, translations with obviously ungrammatical constructions such as (1.b) can result. For unseen n-grams, one would like to model adjectives as being likely to precede nouns in English, for example.

A straightforward approach to address this is to exploit the part-of-speech (POS) tags of the target words during translation (Kirchhoff and Yang, 2005). Though models exploiting POS information are not expressive enough to model long-distance dependencies, they can account for locally ungrammatical constructions such as (1.b). Several attempts have been made to interpolate POS language models

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 869–879,
Edinburgh, Scotland, UK, July 27–31, 2011.

with surface models. Under constrained data conditions, this can lead to improvements. But once larger amounts of training data are used, the gains obtained from adding POS language models decline substantially. This raises the question of why POS language models are not more effective. We argue that one of the short-comings of previous approaches to using POS language models is that these models are estimated globally, not lexically anchored, and hence rather context insensitive.

In this paper, we introduce a novel approach that builds and uses individual, local POS language models for each word in the vocabulary. Our experiments show that it leads to statistically significant improvements over a competitive baseline, using lexicalized reordering and a sizable 5-gram word language model, as well as a standard 7-gram POS language model approach.

2 Part-of-Speech Language Models

2.1 Background

Typically, POS language models are used like word-based language models. N-grams are extracted from a POS-tagged corpus and an n-gram language model is built from that. While word-based models estimate the probability of a string of m words by Equation 2, POS-based models estimate the probability of string of m POS tags by Equation 3.

$$p(w_1^m) \propto \prod_{i=1}^{m} p(w_i|w_{i-n+1}^{i-1}) \quad (2)$$

$$p(t_1^m) \propto \prod_{i=1}^{m} p(t_i|t_{i-n+1}^{i-1}) \quad (3)$$

where, n is the order of the language model, and w_i^j refers to the sub-sequence of words (or tags) from positions i to j.

Word language models can be built directly from large text corpora, such as LDC's Gigaword corpus, but POS models require texts that are annotated with POS tags. Ideally, one would use manually annotated corpora such as the Penn Treebank (Marcus et al., 1993), but since those tend to be small, most approaches rely on larger corpora which have been automatically annotated by a POS tagger or a parser (Koehn et al., 2008). Though automated annotation inevitably contains errors, it is assumed that this is ameliorated by the increased size of annotated data.

The event space of a language models is of size $|V|^n$, where V is the vocabulary, and n is the order of the language model. The vocabulary of POS models, (typically ranging between 40 and 100 tags), is much smaller than the vocabulary of a word model, which can easily approach a million words. Nevertheless, most POS language modeling approaches apply some form of smoothing to account for unseen events (Bonneau-Maynard et al., 2007).

To deploy POS language models in machine translation, translation candidates need to be annotated with POS tags. Each target phrase $\bar{e}$ in a phrase pair $(\bar{f}, \bar{e})$ can be associated with a number of POS tag sequences $\bar{t}_{\bar{e}}$. Heeman (1998) shows that using the joint probability leads to improved perplexity for POS models. For machine translation one can sum over all possible tag sequences, as in Equation 4.

$$p(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \sum_{\mathbf{t}} p(\mathbf{e}, \mathbf{t}|\mathbf{f}) \quad (4)$$

Summing over all possible tag sequences has the disadvantage that it requires one to keep this information during decoding. Below, we opt for an approximate solution, where each target phrase is annotated with the most likely POS tag sequence given the source and target phrase: $\bar{t}_{\bar{e}} = \arg\max_{\bar{t}} p(\bar{t}|\bar{e}, \bar{f})$.

2.2 Effectiveness of POS Language Models

Reported results on the effectiveness of POS language models for machine translation are mixed, in particular when translating into languages that are not morphologically rich, such as English. While they rarely seem to hurt translation quality, there does not seem to be a clear consensus that they significantly improve quality either.

Koehn and Hoang (2007) have reported an increase of 0.86 BLEU points for German-to-English translation for small training data. After relaxing phrase-matching to include lemma and morphological information on the source side, POS language models lead to a decrease of -0.42 BLEU points. Supertagging encapsulates more contextual information than POS tags and Birch et al. (2007) report improvements when comparing a supertag language model to a baseline using a word language model

only. Once the baseline incorporates lexicalized distortion (Tillmann, 2004; Koehn et al., 2005), these improvements disappear. Factored language models have not resulted in significant improvements either. Kirchhoff and Yang (2005) report slight improvements when re-ranking the n-best lists of their decoder, which word tri-grams. But these improvements are less than those gained by re-ranking the n-best lists with a 4-gram word language model.

The impact of POS language models depends among other things on the size of the parallel corpus, the size and order of the word language model, and whether lexicalized distortion models are used. To gauge the potential effectiveness of POS language models without taking into consideration all these factors, we isolate the contribution of the language model by simulating machine translation output using English data only (Al-Onaizan and Papineni, 2006; Post and Gildea, 2008). Taking a set of POS-tagged reference translations of the MT04 Arabic-to-English test set, each English sentence is randomly chunked into n-grams of average length three. The chunks of each sentence, with their corresponding POS tags, are randomly reordered. This is repeated 500 times for each sentence in the test set. The smoothed sentence BLEU score (ignoring brevity penalty) is computed for each reordered sentence with respect to all reference translations. The higher the BLEU score, the more well-formed the reordering is. As each reordered sentence only contains words from at least one of the reference translations, the uni-gram precision is always 1.0. The language model probability is then computed for each reordering. Table 1 shows the average correlations between language model probabilities and BLEU scores.

We can see that the surface language model correlates moderately well with BLEU, explaining about 49% ($r^2 = 0.49$) of the variation, whereas the POS language model does not correlate with BLEU at all.[1] On the other hand, local language models alone (as introduced in Section 3) correlate with BLEU only slightly worse than surface models. The highest correlation is seen when they are interpolated with word models. The BLEU scores in Table 1 are computed using the 1-best sentences after reranking. These system-agnostic correlation results look promising for our local models and the end-to-end translation results in Section 5 confirm these initial findings.

LM	Kendall's τ	Pearson r	BLEU[%]
wordLM	0.53	0.71	80.20
POS 7gLM	0.01	0.01	48.44
locLM	0.45	0.62	76.03
λwordLM+$(1-\lambda)$locLM ($\lambda = 0.92$)	0.54	0.73	80.98

Table 1: Correlation between randomly permuted English reference translations and BLEU.

3 Local Language Models

In this section, we introduce a novel approach to language modeling that is more context-sensitive than standard POS language models. Instead of using one global POS language model that is built by using all of a mono-lingual corpus in the target language, we build individual models, or local models, for each word-POS pair using the POS tags surrounding each occurrence of that pair. This adds an aspect of lexicalization that is entirely absent in previous POS language models. The effect is that the resulting n-gram probability distributions of each local model are more biased towards the contextual constraints of each individual word-POS pair. This is similar to the idea of cached language models (Kuhn, 1988), but more fine-grained and with a tighter integration of POS and lexical information.

3.1 Definition of Local Language Models

Each conditional probability of order n in a local model for the word-POS pair $w:t$ is of the form:

$$p_{w:t}(t_n, p_n | t_1 : p_1, \ldots, t_{n-1} : p_{n-1})$$

where t_i refers to POS tags and p_i to positions relative to an occurrence of the pair $(w:t)$. For example, consider the sentence fragment in Figure 1. The conditional local n-gram probabilities (a–d) are generated from the occurrence of the word *told* with POS tag VBD. Probability (c) in Figure 1 estimates that a word with POS tag NN occurs two positions to the right of *told*, given the n-gram history that a noun occurs to its left and a determiner to its right.

[1] Interpolating both models does not lead to further correlation improvements.

position	...	11	12	13	14	15	16	17	...
relative position	...	-3	-2	-1	0	+1	+2	+3	...
word	...	the	new	mayor	told	the	reporter	to	...
POS	...	DT	JJ	NN	VBD	DT	NN	TO	...

(a) $p_{told:\mathrm{VBD}}(\mathrm{NN{:}-1}|\mathrm{DT{:}-3\ JJ{:}-2})$

(b) $p_{told:\mathrm{VBD}}(\mathrm{DT{:}+1}|\mathrm{JJ{:}-2\ NN{:}-1})$

(c) $p_{told:\mathrm{VBD}}(\mathrm{NN{:}+2}|\mathrm{NN{:}-1\ DT{:}+1})$

(d) $p_{told:\mathrm{VBD}}(\mathrm{TO{:}+3}|\mathrm{DT{:}+1\ NN{:}+2})$

Figure 1: Sentence fragment with the tri-gram probabilities (a–d) linked to *told*.

For each local model we use a sliding window considering all n-grams of length n starting n words to the left and ending n words to the right of an occurrence of the word-POS pair of the model at hand.

All local model probabilities are smoothed using Witten-Bell smoothing and interpolation.[2] POS tags are annotated with positional information to distinguish between lower-order estimates such as $p_{told:\mathrm{VBD}}(\mathrm{NN{+}2})$ and $p_{told:\mathrm{VBD}}(\mathrm{NN{+}3})$ both of which can arise when backing off during smoothing. Without positional information, $p_{told:\mathrm{VBD}}(\mathrm{NN})$ only estimates the probability of the tag NN occurring within the proximity of *told*.[3]

A local model of order n contains the conditional probabilities for words occurring at relative positions -1, +1, ...+n. Therefore the probability of a word occurrence is estimated by all local models covering this word's position. Figure 2 shows schematically how overlapping n-gram probabilities interact. E.g., the probability of word w_{i+2} is based on the probability of the local model for w_{i+1}, w_i, w_{i-1}, and w_{i-2} (the last two are not shown in Figure 2 for space reasons). Formally, the conditional probability of a word-POS pair, given its word and POS tag history is defined in Equation 5.

$$p(w_i, t_i \mid w_{i-n+1}^{i-1}, t_{i-n+1}^{i-1}) = p_{w_i:t_i}(t_{i-1}\text{:-1} \mid \langle t_{i-n}\text{:-}n, \ldots, t_{i-2}\text{:-2}\rangle) \cdot \prod_{j=0}^{n-1} p_{w_{i-n+j}:t_{i-n+j}}(t_i\text{:}n-j \mid H_{i,n}[j,\cdot]) \quad (5)$$

[2]The smaller event space of local models often leads to incomplete counts-of-counts, preventing the use of Kneser-Ney smoothing (Chen and Goodman, 1999).

[3]Despite the notational similarities, our approach should not be confused with projected POS models, which use source side POS tags to model reordering (Och et al., 2004).

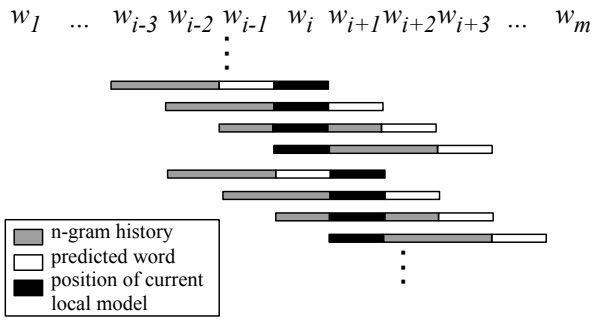


Figure 2: Schema of overlapping local language model applications.

where $H_{i,n}$ is an $n \times n$ matrix specifying the history of the word at position i. Each row j of $H_{i,n}$ represents the history of the conditional probability belonging to the local model associated with position $i-n+j$. Each entry $H_{i,n}[j,k]$ is defined as follows:

$$H_{i,n}[j,k] = \begin{cases} t_{i-n+k}\text{:}k-j & \text{if } j \neq k \\ \epsilon & \text{otherwise} \end{cases}$$

where t_{i-n+k} is the POS tag at position $i-n+k$ and $k-j$ is the relative position with respect to the diagonal of $H_{i,n}$, i.e., the position of the local language model corresponding to row j. $H_{i,n}[j,\cdot]$ is the jth row vector from which the jth entry (the empty element) has been removed. For instance, given the example in Figure 1, $H_{14,3}$ is

$$H_{14,3} = \begin{bmatrix} \epsilon & \mathrm{JJ{:}+1} & \mathrm{NN{:}+2} \\ \mathrm{DT{:}-1} & \epsilon & \mathrm{NN{:}+1} \\ \mathrm{DT{:}-2} & \mathrm{JJ{:}-1} & \epsilon \end{bmatrix}$$

For convenience we assume that the row and column indices are 0-based, i.e., the upper-left entry of a matrix is referred to by $H_{i,n}[0,0]$. In this example, $H_{14,3}[1,\cdot] = \langle \mathrm{DT{:}-1},\ \mathrm{NN{:}+1} \rangle$.

position	0	1	2	3	4	5	6
token	<s>	cuba	frees	more	dissidents	.	</s>
POS tag	<s>	NNP	VBZ	JJR	NNS	.	</s>

$$
\begin{aligned}
p(\text{cuba, NNP}|w_0^0, t_0^0) &= p_{\text{cuba:NNP}}(\text{<s>:-1}) \cdot p_{\text{<s>:<s>}}(\text{NNP:+1}) \\
p(\text{frees, VBZ}|w_0^1, t_0^1) &= p_{\text{frees:VBZ}}(\text{NNP:-1}|\text{<s>:-2}) \cdot p_{\text{<s>:<s>}}(\text{VBZ:+2}|\text{NNP:+1}) \\
&\quad \cdot p_{\text{cuba:NNP}}(\text{VBZ:+1}|\text{<s>:-1}) \\
p(\text{more, JJR}|w_0^2, t_0^2) &= p_{\text{more:JJR}}(\text{VBZ:-1}|\text{NNP:-3 VBZ:-2}) \cdot p_{\text{<s>:<s>}}(\text{JJR:+3}|\text{NNP:+1 VBZ:+2}) \\
&\quad \cdot p_{\text{cuba:NNP}}(\text{JJR:+2}|\text{<s>:-1 VBZ:+1}) \cdot p_{\text{frees:VBZ}}(\text{JJR:+1}|\text{<s>:-2 NNP:-1}) \\
p(\text{dissidents, NNS}|w_1^3, t_1^3) &= p_{\text{dissidents:NNS}}(\text{JJR:-1}|\text{NNP:-3 VBZ:-2}) \cdot p_{\text{cuba:NNP}}(\text{NNS:+3}|\text{VBZ:+1 JJR:+2}) \\
&\quad \cdot p_{\text{frees:VBZ}}(\text{NNS:+2}|\text{NNP:-1 JJR:+1}) \cdot p_{\text{more:JJR}}(\text{NNS:+1}|\text{NNP:-2 VBZ:-1}) \\
p(.\,, .|w_2^4, t_2^4) &= p_{.:.}(\text{NNS:-1}|\text{VBZ:-3 JJR:-2}) \cdot p_{\text{frees:VBZ}}(\text{.:+3}|\text{JJR:+1 NNS:+2}) \\
&\quad \cdot p_{\text{more:JJR}}(\text{.:+2}|\text{VBZ:-1 NNS:+1}) \cdot p_{\text{dissidents:NNS}}(\text{.:+1}|\text{VBZ:-2 JJR:-1}) \\
p(\text{</s>, </s>}|w_3^5, t_3^5) &= p_{\text{</s>:</s>}}(\text{.:-1}|\text{JJR:-3 NNS:-2}) \cdot p_{\text{more:JJR}}(\text{</s>:+3}|\text{NNS:+1 .:+2}) \\
&\quad \cdot p_{\text{dissidents:NNS}}(\text{</s>:+2}|\text{JJR:-1 .:+1}) \cdot p_{.:.}(\text{</s>:+1}|\text{JJR:-2 NNS:-1})
\end{aligned}
$$

Figure 3: Language model probability computation for the sentence "Cuba frees more dissidents." using our local language modeling approach.

The example in Figure 3 shows word-by-word how tri-gram local language models are used to compute the probability of a whole sentence.

Our local language model approach also bears some resemblance to statistical approaches to modeling subcategorization frames (Manning, 1993). While our approach is more general by considering all words and not just focusing on verbal subcategorization frames, it is also more shallow in the sense that only part-of-speech categories are considered which does not model any contextual relationships on the phrase level.

3.2 Building Local Language Models

To build the local language models, we use the SRILM toolkit (Stolcke, 2002), which is commonly applied in speech recognition and statistical machine translation. While SRILM collects n-gram statistics from all n-grams occurring in a corpus to build a single global language model, we build a language model for each word-POS pair only using the n-grams within the proximity of occurrences for that word-POS pair in a POS-tagged corpus. This results in separate n-gram count files, which are then processed by SRILM to build the individual language models.[4] Charniak's parser (Charniak, 2000) is used to POS tag the corpus.

[4]The pre-processing scripts are available at `http://www.science.uva.nl/~christof/locLM/`.

3.3 Decoder Integration

Several approaches that integrate POS language models have focused on n-best list re-ranking only (Hasan et al., 2006; Wang et al., 2007). Often this is due to the computational (and implementational) complexities of integrating more complex language models with the decoder, although it is expected that a tighter integration with the decoder itself leads to better improvements than n-best list re-ranking.

Integrating our local language modeling approach with a decoder is straightforward. Our baseline decoder already uses SRILM's API for computing word language model probabilities. Since SRILM supports arbitrarily many language models, local language models can be added using the same functionalities of SRILM's API. For the experiments discussed in Section 4, we add about 150,000 local language models to the word model. All local language model probabilities are coupled with the same feature weight. Potentially, improvements could be gained from using separate weights for individual local models, but this would require an optimization procedure such as MIRA (Chiang et al., 2009), which can handle a larger number of features.

During decoding no POS tagging ambiguities are resolved. Each target phrase is associated with its most likely POS tag sequence, given the source and target side of the phrase pair; see Section 2.1.

4 Experimental Setup

Three approaches are compared in our experiments: the baseline system is a phrase-based statistical machine translation system (Koehn et al., 2003), very similar to Moses (Koehn et al., 2007), using a word-based 5-gram language model. The second approach extends the baseline by including a 7-gram POS-based language model. The third approach represents the work described in this paper, extending the baseline by including 4-gram local language models.

Translation quality is evaluated for two language pairs: Arabic-to-English and Chinese-to-English. NIST's MT-Eval test sets are used for both pairs. Only resources allowed under NIST's constrained data conditions are used to train the language, translation, and lexicalized distortion models.

To see whether our local language models result in improvements over a competitive baseline, we designed the baseline to use a large 5-gram word language model and lexicalized distortion modeling, both of which are known to cancel-out improvements gained from POS language models (Birch et al., 2007; Kirchhoff and Yang, 2005). The 5-gram word language model is trained on the Xinhua and AFP sections of the Gigaword corpus (3rd edition, LDC2007T40) and the target side of the bitext. We removed from the training data all documents released during the periods that overlap with the publication dates of the documents included in our development or test data sets. In total, 630 million tokens were used to build the word language model. The language model was trained using SRILM with modified Kneser-Ney smoothing and interpolation (Chen and Goodman, 1999). It is common practice not to include higher-order n-grams that occur fewer than a predefined number of times. Here, we applied rather conservative cut-offs, by ignoring 3-, 4-, and 5-grams that occurred only once. The 7-gram POS and 4-gram local language models were both trained on the POS tagged English side of the bitext and 10M sentences from Gigaword's Xinhua and AFP sections.

The data for building the translation models were primarily drawn from the parallel news resources distributed by the Linguistic Data Consortium (LDC).[5] The Arabic-English bitext consists of 11.4M source and 12.6M target tokens, and the Chinese-English bitext of 10.6M source and 12.3M target tokens. Word alignment was performed running GIZA++ in both directions and generating the symmetric alignments using the 'grow-diag-final-and' heuristics.

All three approaches, including the baseline, use lexicalized distortion, distinguishing between monotone, swap, and discontinuous reordering, all with respect to the previous and next phrase (Koehn et al., 2005). The distortion limit is set to 5 for Arabic-to-English, and 6 for Chinese-to-English. For each source phrase the top 30 translations are considered.

For tuning and testing we use NIST's official MT-Eval test sets. MT04 was used as the development set for both language pairs. Testing was carried out on MT05 to MT09 for Arabic-English and MT05 to MT08 for Chinese-English. NIST did not release a new Chinese-English test set for MT-Eval 2009. Parameter tuning of the decoder was done with minimum error rate training (MERT) (Och, 2003), adapted to BLEU maximization.

As evaluation metrics we used NIST's adaptation of BLEU-4 (Papineni et al., 2001), version 13a, where the brevity penalty is based on the reference translation with the closest length, and translation error rate (TER) version 0.7.25 (Snover et al., 2006). All results reported here are case-insensitive. TER scores are shown as 1-TER.

To see whether the differences between the approaches we compared in our experiments are statistically significant, we apply approximate randomization (Noreen, 1989); Riezler and Maxwell (2005) have shown that approximate randomization is less sensitive to Type-I errors, i.e., less likely to falsely reject the null hypothesis, than bootstrap resampling (Koehn, 2004) in the context of machine translation.

5 Results and Analysis

The Arabic-to-English results are shown in Table 2, and the Chinese-to-English results in Table 3. All results are subdivided by genre following NIST's genre classification. Note that MT06 con-

[5] LDC catalog numbers for Arabic-English: LDC2004E72, LDC2004T17, LDC2004T18, LDC2005E46, LDC2005E83, LDC2006E25, LDC2006E34, LDC2006E85, LDC2006E92, and LDC2007T08. For Chinese-English: LDC2002E18, LDC2003E07, LDC2003E14, LDC2005E83, LDC2005T06, LDC2006E34, LDC2006E85, and LDC2006E92.

systems and improvements		MT04 tune	MT05 NW	MT06 NW	MT06 WB	MT06 ALL	MT08 NW	MT08 WB	MT08 ALL	MT09 NW	MT09 WB	MT09 ALL	MT05–09 NW	MT05–09 WB	MT05–09 ALL
BLEU[%]															
1a	wordLM	51.90	53.83	46.76	34.69	43.41	48.77	33.26	42.37	52.97	34.25	44.34	50.51	34.00	45.63
2a	+posLM	51.92	54.29	47.02	34.44	43.51	48.81	33.30	42.31	53.52	34.04	44.36	50.89	33.87	45.70
3a	> wordLM	+0.02	+0.46$^{\blacktriangle}$	+0.26	−0.25	+0.10	+0.04	+0.04	−0.06	+0.55$^{\blacktriangle}$	−0.21	+0.02	+0.38$^{\blacktriangle}$	−0.13	+0.07
4a	+locLM	**52.65**	**55.08**	**47.24**	**35.17**	**43.88**	**49.61**	**33.67**	**42.92**	**54.39**	**34.40**	**44.82**	**51.57**	**34.33**	**46.22**
5a	> wordLM	+0.75$^{\blacktriangle}$	+1.25$^{\blacktriangle}$	+0.48$^{\blacktriangle}$	+0.48$^{\triangle}$	+0.47$^{\blacktriangle}$	+0.84$^{\blacktriangle}$	+0.41	+0.55$^{\blacktriangle}$	+1.42$^{\blacktriangle}$	+0.15	+0.48$^{\blacktriangle}$	+1.06$^{\blacktriangle}$	+0.33$^{\triangle}$	+0.59$^{\blacktriangle}$
6a	> +posLM	+0.73$^{\blacktriangle}$	+0.79$^{\blacktriangle}$	+0.22	+0.73$^{\blacktriangle}$	+0.37$^{\triangle}$	+0.80$^{\blacktriangle}$	+0.37	+0.61$^{\blacktriangle}$	+0.87$^{\blacktriangle}$	+0.36	+0.46$^{\blacktriangle}$	+0.68$^{\blacktriangle}$	+0.46$^{\blacktriangle}$	+0.52$^{\blacktriangle}$
1-TER[%]															
1b	wordLM	58.32	59.04	54.27	45.62	51.68	55.59	44.41	50.69	59.90	46.43	53.03	56.94	45.49	53.13
2b	+posLM	58.54	59.72	**54.90**	**45.67**	**52.14**	55.75	44.64	50.89	60.49	46.72	53.47	57.46	45.70	53.55
3b	> wordLM	+0.22$^{\triangle}$	+0.68$^{\blacktriangle}$	+0.63$^{\blacktriangle}$	+0.05	+0.46$^{\blacktriangle}$	+0.16	+0.23	+0.20$^{\triangle}$	+0.59$^{\blacktriangle}$	+0.29$^{\triangle}$	+0.44$^{\blacktriangle}$	+0.52$^{\blacktriangle}$	+0.21$^{\blacktriangle}$	+0.42$^{\blacktriangle}$
4b	+locLM	**58.95**	**60.06**	54.88	45.62	52.11	**56.42**	**44.91**	**51.38**	**60.91**	**46.84**	**53.74**	**57.79**	**45.83**	**53.81**
5b	> wordLM	+0.63$^{\blacktriangle}$	+1.02$^{\blacktriangle}$	+0.61$^{\blacktriangle}$	+0.00	+0.43$^{\blacktriangle}$	+0.83$^{\blacktriangle}$	+0.50$^{\blacktriangle}$	+0.69$^{\blacktriangle}$	+1.01$^{\blacktriangle}$	+0.41$^{\triangle}$	+0.71$^{\blacktriangle}$	+0.85$^{\blacktriangle}$	+0.34$^{\blacktriangle}$	+0.68$^{\blacktriangle}$
6b	> +posLM	+0.41$^{\blacktriangle}$	+0.34$^{\triangle}$	−0.02	−0.05	−0.03	+0.67$^{\blacktriangle}$	+0.27	+0.49$^{\blacktriangle}$	+0.42$^{\triangle}$	+0.12	+0.27$^{\triangle}$	+0.33$^{\blacktriangle}$	+0.13	+0.26$^{\blacktriangle}$
# segments		1,353	1,056	1,033	764	1,797	813	547	1,360	586	727	1,313	3,488	2,038	5,526

Table 2: Results for Arabic-to-English translation. Comparison of our approach (**+locLM**, rows 4a/b) to the baseline using a word language model (**wordLM**, rows 1a/b) and a competing approach using a POS-based language model (**+posLM**, rows 2a/b). Results are presented using BLEU[%] (rows 1a–6a) and 1-TER[%] (rows 1b–6b) and broken down by genre: NW=newswire, WB=web, and ALL=NW∪WB. Rows 3a/b, 5a/b, and 6a/b show the relative improvements over the system mentioned to the right of the $>$ sign. Statistically significant improvements/declines (using approximate randomization) at the $p < .01$ level are marked $^{\blacktriangle}$/$^{\blacktriangledown}$ and $^{\triangle}$/$^{\triangledown}$ at the $p < .05$ level.

tains the genres 'broadcast news' and 'newsgroup'. In both tables, the former has been classified under 'newswire' and the latter under 'web'.

The first approach is the baseline system 'wordLM' (rows 1a/b in Tables 2 and 3), which uses a 5-gram word-based language model. The next approach '+posLM' extends the baseline by adding a 7-gram POS language model (rows 2a/b in both tables). Rows 3a/b show the relative improvements over the baseline. The third approach '+locLM' (rows 4a/b) uses local language models in addition to the baseline's word-based model. Note that +locLM does not use the 7-gram POS language model as well. Rows 5a/b show the relative improvements of the local modeling approach over the baseline and rows 6a/b the improvements over the approach using a POS language model.

Let us first take a closer look at the Arabic-to-English results in Table 2. The approach using a POS language model results in statistically significant improvements for only one test set (MT05) and the newswire documents of MT09. The average improvements across all sets and genres are negligible ($+0.07$ BLEU). Our local language modeling approach achieves the highest BLEU scores for all test sets and across all genres. In particular, the improvements of +1.06 BLEU for newswire documents are substantial. With the exception of MT08-WB and MT09-WB all BLEU improvements over the baseline are statistically significant.

When evaluating with 1-TER, local language modeling also achieves the best results, with the exception of MT06, where the POS language model approach performs slightly better.

Turning to the Chinese-English results in Table 3, we see similar improvements in BLEU. The improvements of using a POS language model are negligible ($+0.04$ BLEU). Here as well, local language modeling leads to the best results, with substantial improvements of +0.88 BLEU for web documents.

The major difference between Arabic-English and Chinese-English is the discrepancy between BLEU score improvements and decreases in 1-TER. While we cannot explain this discrepancy, it is worth noting that similar discrepancies between BLEU and TER and Arabic-to-English and Chinese-to-English translation can be found in the literature. The results described in Shen et al. (2009) show a strong correlation between BLEU and 1-TER improvements[6]

[6]Shen et al. (2009) report TER rather than 1-TER scores.

systems and improvements		MT04 tune	MT05 NW	MT06 NW	MT06 WB	MT06 ALL	MT08 NW	MT08 WB	MT08 ALL	MT05–08 NW	MT05–08 WB	MT05–08 ALL
BLEU[%]												
1a	wordLM	37.32	32.55	33.33	23.40	31.16	28.67	17.57	24.03	31.93	19.82	29.30
2a	+posLM	37.32	32.47	33.13	23.67	31.06	28.63	**18.46**	24.35	31.82	20.46	29.34
3a	> wordLM	+0.00	−0.08	−0.20	+0.27	−0.10	−0.04	+0.89$^{\blacktriangle}$	+0.32	−0.11	+0.64$^{\blacktriangle}$	+0.04
4a	+locLM	**38.15**	**33.05**	**33.33**	**24.62**	**31.42**	**29.52**	18.24	**24.79**	**32.36**	**20.70**	**29.82**
5a	> wordLM	+0.83$^{\blacktriangle}$	+0.50$^{\triangle}$	+0.00	+1.22$^{\blacktriangle}$	+0.26	+0.85$^{\blacktriangle}$	+0.67$^{\triangle}$	+0.76$^{\blacktriangle}$	+0.43$^{\blacktriangle}$	+0.88$^{\blacktriangle}$	+0.52$^{\blacktriangle}$
6a	> +posLM	+0.83$^{\blacktriangle}$	+0.58$^{\blacktriangle}$	+0.20	+0.95$^{\blacktriangle}$	+0.36$^{\triangle}$	+0.89$^{\blacktriangle}$	−0.22	+0.44$^{\triangle}$	+0.54$^{\blacktriangle}$	+0.24	+0.48$^{\blacktriangle}$
1-TER[%]												
1b	wordLM	**42.81**	**40.73**	**42.99**	**39.42**	**42.15**	**40.42**	**36.77**	**38.78**	**41.53**	**37.77**	**40.63**
2b	+posLM	42.50	40.60	42.75	38.87	41.84	39.76	36.75	38.41	41.23	37.55	40.34
3b	> wordLM	−0.31$^{\triangledown}$	−0.13	−0.24	−0.55	−0.31$^{\triangledown}$	−0.66$^{\blacktriangledown}$	−0.02	−0.37$^{\triangledown}$	−0.30$^{\blacktriangledown}$	−0.22	−0.29$^{\blacktriangledown}$
4b	+locLM	42.77	40.49	42.62	39.40	41.86	40.00	36.11	38.26	41.20	37.35	40.27
5b	> wordLM	−0.04	−0.24	−0.37	−0.02	−0.29	−0.42	−0.66$^{\blacktriangledown}$	−0.52$^{\blacktriangledown}$	−0.33$^{\triangledown}$	−0.42$^{\triangledown}$	−0.36$^{\blacktriangledown}$
6b	> posLM	+0.27	−0.11	−0.13	+0.53	+0.02	+0.24	−0.64$^{\blacktriangledown}$	−0.15	−0.03	−0.20	−0.07
# segments		1,788	1,082	1,181	483	1,664	691	666	1,357	2,954	1,149	4,103

Table 3: Comparison of our system for Chinese-to-English translation. See Table 2 for details on notation.

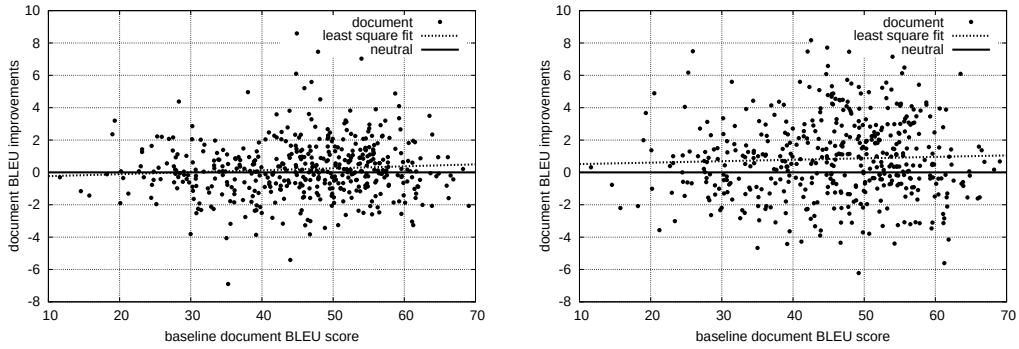
for Arabic-to-English on the MT06 and MT08 sets, but for Chinese-to-English the correlation seems to be much weaker and BLEU improvements of +0.75 can correspond to decreases of up to -0.80 in 1-TER.

One of the motivations of using POS language models in general, and local language models in our case, is to improve the fluency of translations, which should be reflected in increased precision for higher-order n-grams. Table 4 shows that this is the case when comparing local modeling to both word and POS language models for Arabic-to-English translation. The same trend, but to a somewhat weaker degree can be observed for Chinese-to-English.

	Prec-1	Prec-2	Prec-3	Prec-4	BP
Arabic-English (MT05–09)					
wordLM	81.38	54.51	38.10	26.99	0.987
+posLM	81.81	54.82	38.34	27.17	0.983
+locLM	81.90	55.35	39.01	27.86	0.981
Chinese-English (MT05–08)					
wordLM	75.03	40.56	22.55	12.93	0.955
+posLM	74.81	40.30	22.41	12.83	0.962
+locLM	74.24	40.70	22.83	13.19	0.966

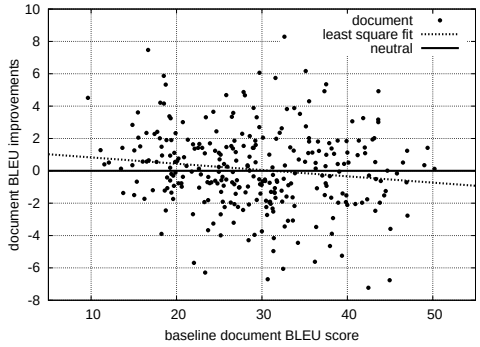
Table 4: BLEU n-gram precision ($1 \leq n \leq 4$) and Brevity Penalty (BP) scores over all test sets.

The effectiveness of a POS language model often diminishes with improved translation quality of the base system to which it is added. Naturally, we are interested in the extent that this diminishing effect also holds for our local language models. A full experimental setup, varying all relevant factors, such as language, translation, and distortion model size, and the various meta-parameters, is beyond the scope of this paper. Nevertheless, we can gauge this by taking a closer look at the distribution of improvements within our experiments. Figure 4 shows performance improvements in document-level BLEU for both language pairs. The document-level BLEU score for the baseline system is plotted on the x-axis and improvements are plotted on the y-axis. The dotted line is the linear fit (using least square regression). If the effectiveness of either added model (POS or local) diminishes with increasing translation quality, we would expect a declining regression line. This is not the case for Arabic-to-English translation. Relative improvements for both added models increase as the translation quality of the baseline increases. The slope of both regression fits is almost identical, but the y-intercept is larger for our local modeling approach. Note that the small slope is also due to difference in scale between full BLEU scores and relative improvements. We can observe the opposite for Chinese-to-English translation, where the slope is negative. Both models seem to help more for documents with lower baseline translation quality. For the POS model, the regression line intersects with the neutral line (± 0 improvement) at around 31 BLEU, which is close to the average BLEU score and in line with its negligible improvements (see Ta-

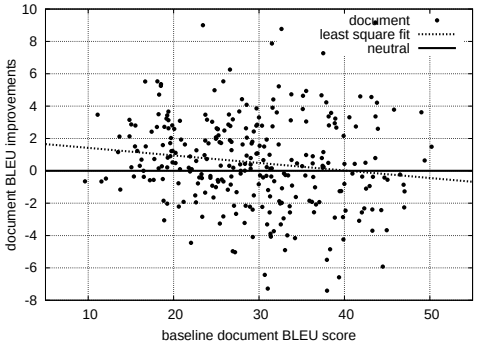


Arabic-English: +posLM > wordLM

Arabic-English: +locLM > wordLM



Chinese-English: +posLM > wordLM



Chinese-English: +locLM > wordLM

Figure 4: Correlation between baseline BLEU scores for individual documents and the relative, absolute improvements achieved by +posLM (left) and +locLM (right). BLEU scores (and improvements) are computed at the document level.

ble 3). For the local language model, the regression line intersects with the neutral line at about 40 BLEU, suggesting that until translation quality improves substantially, local language models could still have a positive impact.

6 Related Work

The main goal of this paper is to show that by tying POS language models to lexical items, we get more accurate distributions for specific words. The work on factored language models (Bilmes and Kirchhoff, 2003) is related to our work to the extent that it also mixes POS tags with lexical information, albeit in a very different manner. Factored language models use more general representations, such as POS tags or stems, only during back-off. Kirchhoff and Yang (2005) applied factored language models to machine translation but the improvements were negligible.

Collins et al. (2005) proposed a discriminative language modeling approach that uses mixtures of POS and surface information and showed that it leads to a reduction in speech recognition word error rates. On the other hand, their approach seems more suited for n-best list re-ranking and it is not clear whether those improvements carry over to machine translation. Li and Khudanpur (2008) adapted this discriminative approach to machine translation re-ranking but used surface forms only.

Wang et al. (2007) and Zheng et al. (2008) use elaborately enriched representations, called *super abstract role values* (Wang and Harper, 2002), which capture contextual dependencies using lexical categories, role labels, and dependency grammar structures. So far their approach has been limited to re-ranking n-best lists only.

7 Conclusion

Though POS language models do not lead to significant improvements over a competitive baseline, we have shown that a competitive phrase-based baseline system can benefit from using POS information by building lexically anchored local models. Our local model approach does not only lead to more context-specific probability distributions, but also takes ad-

vantage of the language model probability of each word being based on all surrounding local models. The evaluations for Arabic- and Chinese-to-English show that local models lead to statistically significant improvements across different test sets and genres. Correlating the translation quality of the baseline with the improvements that result from adding local models, further suggests that these improvements are sustainable and should carry over to improved baseline systems.

Acknowledgments

This research was funded in part by the European Commission through the CoSyne project FP7-ICT-4-248531, the European Commission's ICT Policy Support Program as part of the Competitiveness and Innovation Framework Program, CIP ICT-PSP under grant agreement nr. 250430, and the PROMISE Network of Excellence co-funded by the 7th Framework Programme of the European Commission, grant agreement no. 258191.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 529–536.

Jeff A. Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of the the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 4–6.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16.

Hélène Bonneau-Maynard, Alexandre Allauzen, Daniel Déchelotte, and Holger Schwenk. 2007. Combining morphosyntactic enriched representation with n-best reranking in statistical translation. In *Proceedings of the NAACL-HLT Workshop on Syntax and Structure in Statistical Translation*, pages 65–71.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–393.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 218–226.

Michael Collins, Brian Roark, and Murat Saraclar. 2005. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 507–514.

Saša Hasan, Oliver Bender, and Hermann Ney. 2006. Reranking translation hypotheses using structural properties. In *Proceedings of the EACL Workshop on Learning Structured Information in Natural Language Applications*, pages 41–48.

Peter Heeman. 1998. POS tagging versus classes in language modeling. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 179–187.

Katrin Kirchhoff and Mei Yang. 2005. Improved language modeling for statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 125–128.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Inter-*

active Poster and Demonstration Sessions, pages 177–180.

Philipp Koehn, Abhishek Arun, and Hieu Hoang. 2008. Towards better machine translation quality for the german–english language pairs. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 139–142.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Roland Kuhn. 1988. Speech recognition and the frequency of recently used words: a modified Markov model for natural language. In *Proceedings of the 12th conference on Computational Linguistics*, pages 348–350.

Zhifei Li and Sanjeev Khudanpur. 2008. Large-scale discriminative n-gram language models for statisti- cal machine translation. In *Proceedings of AMTA*, pages 133–142.

Christopher D. Manning. 1993. Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 235–242.

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

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley-Interscience.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, ALex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics*, pages 161–168.

Franz-Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2001)*, pages 311–318.

Matt Post and Daniel Gildea. 2008. Parsers as language models for statistical machine translation. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas*, pages 172–181.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21:492–518.

Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2009. Effective use of linguistic and contextual information for statistical machine translation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 72–80.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Andreas Stolcke. 2002. SRILM—an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL-04)*, pages 101–104.

Wen Wang and Mary P. Harper. 2002. The SuperARV language model: investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 238–247.

Wen Wang, Andreas Stolcke, and Jing Zheng. 2007. Reranking machine translation hypotheses with structured and web-based language models. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 159–164.

Jing Zheng, Necip Fazil Ayan, Wen Wang, Dimitra Vergyri, Nicolas Scheffer, and Andreas Stolcke. 2008. SRI systems in the NIST MT08 Evaluation. In *Proceedings of the NIST 2008 Open MT Evaluation Workshop*.

Fast Generation of Translation Forest for Large-Scale SMT Discriminative Training

Xinyan Xiao, Yang Liu, Qun Liu, and **Shouxun Lin**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{xiaoxinyan,yliu,liuqun,sxlin}@ict.ac.cn

Abstract

Although discriminative training guarantees to improve statistical machine translation by incorporating a large amount of overlapping features, it is hard to scale up to large data due to decoding complexity. We propose a new algorithm to generate translation forest of training data in linear time with the help of word alignment. Our algorithm also alleviates the oracle selection problem by ensuring that a forest always contains derivations that exactly yield the reference translation. With millions of features trained on 519K sentences in 0.03 second per sentence, our system achieves significant improvement by 0.84 BLEU over the baseline system on the NIST Chinese-English test sets.

1 Introduction

Discriminative model (Och and Ney, 2002) can easily incorporate non-independent and overlapping features, and has been dominating the research field of statistical machine translation (SMT) in the last decade. Recent work have shown that SMT benefits a lot from exploiting large amount of features (Liang et al., 2006; Tillmann and Zhang, 2006; Watanabe et al., 2007; Blunsom et al., 2008; Chiang et al., 2009). However, the training of the large number of features was always restricted in fairly small data sets. Some systems limit the number of training examples, while others use short sentences to maintain efficiency.

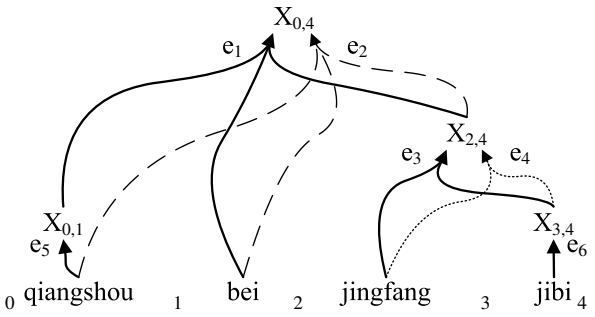
Overfitting problem often comes when training many features on a small data (Watanabe et al., 2007; Chiang et al., 2009). Obviously, using much more data can alleviate such problem. Furthermore, large data also enables us to globally train millions of sparse lexical features which offer accurate clues for SMT. Despite these advantages, to the best of our knowledge, no previous discriminative training paradigms scale up to use a large amount of training data. The main obstacle comes from the complexity of packed forests or n-best lists generation which requires to search through all possible translations of each training example, which is computationally prohibitive in practice for SMT.

To make normalization efficient, contrastive estimation (Smith and Eisner, 2005; Poon et al., 2009) introduce neighborhood for unsupervised log-linear model, and has presented positive results in various tasks. Motivated by these work, we use a *translation forest* (Section 3) which contains both "*reference*" derivations that potentially yield the reference translation and also *neighboring* "*non-reference*" derivations that fail to produce the reference translation.[1] However, the complexity of generating this translation forest is up to $\mathcal{O}(n^6)$, because we still need biparsing to create the reference derivations.

Consequently, we propose a method to fast generate a subset of the forest. The key idea (Section 4) is to initialize a reference derivation tree with maximum score by the help of word alignment, and then traverse the tree to generate the subset forest in linear time. Besides the efficiency improvement, such a forest allows us to train the model without resort-

[1] Exactly, there are no reference derivations, since derivation is a latent variable in SMT. We call them reference derivation just for convenience.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 880–888,
Edinburgh, Scotland, UK, July 27–31, 2011.



hyper-edge		rule
e_1	r_1	$X \Rightarrow \langle X_1$ *bei* X_2, X_1 was $X_2 \rangle$
e_2	r_2	$X \Rightarrow \langle$ *qiangshou bei* X_1, the gunman was $X_1 \rangle$
e_3	r_3	$X \Rightarrow \langle$ *jingfang* X_1, X_1 by the police$\rangle$
e_4	r_4	$X \Rightarrow \langle$ *jingfang* X_1, police $X_1 \rangle$
e_5	r_5	$X \Rightarrow \langle$ *qiangshou*, the gunman$\rangle$
e_6	r_6	$X \Rightarrow \langle$ *jibi*, shot dead$\rangle$

Figure 1: A translation forest which is the running example throughout this paper. The reference translation is "*the gunman was killed by the police*". (1) Solid hyperedges denote a "reference" derivation tree t_1 which exactly yields the reference translation. (2) Replacing e_3 in t_1 with e_4 results a competing non-reference derivation t_2, which fails to swap the order of $X_{3,4}$. (3) Removing e_1 and e_5 in t_1 and adding e_2 leads to another reference derivation t_3. Generally, this is done by deleting a node $X_{0,1}$.

ing to constructing the oracle reference (Liang et al., 2006; Watanabe et al., 2007; Chiang et al., 2009), which is non-trivial for SMT and needs to be determined experimentally. Given such forests, we globally learn a log-linear model using stochastic gradient descend (Section 5). Overall, both the generation of forests and the training algorithm are scalable, enabling us to train millions of features on large-scale data.

To show the effect of our framework, we globally train millions of word level context features motivated by word sense disambiguation (Chan et al., 2007) together with the features used in traditional SMT system (Section 6). Training on 519K sentence pairs in 0.03 seconds per sentence, we achieve significantly improvement over the traditional pipeline by 0.84 BLEU.

2 Synchronous Context Free Grammar

We work on synchronous context free grammar (SCFG) (Chiang, 2007) based translation. The elementary structures in an SCFG are rewrite rules of the form:

$$X \Rightarrow \langle \gamma, \alpha \rangle$$

where γ and α are strings of terminals and nonterminals. We call γ and α as the source side and the target side of rule respectively. Here a rule means a phrase translation (Koehn et al., 2003) or a translation pair that contains nonterminals.

We call a sequence of translation steps as a *derivation*. In context of SCFG, a derivation is a sequence of SCFG rules $\{r_i\}$. *Translation forest* (Mi et al., 2008; Li and Eisner, 2009) is a compact representation of all the derivations for a given sentence under an SCFG (see Figure 1). A *tree* t in the forest corresponds to a derivation. In our paper, tree means the same as derivation.

More formally, a **forest** is a pair $\langle V, E \rangle$, where V is the set of **nodes**, E is the set of **hyperedge**. For a given source sentence $\mathbf{f} = f_1^n$, Each node $v \in V$ is in the form $X_{i,j}$, which denotes the recognition of nonterminal X spanning the substring from the i through j (that is $f_{i+1}...f_j$). Each hyperedge $e \in E$ connects a set of antecedent to a single consequent node and corresponds to an SCFG rule $r(e)$.

3 Our Translation Forest

We use a translation forest that contains both "*reference*" derivations that potentially yield the reference translation and also some neighboring "*non-reference*" derivations that fail to produce the reference translation. Therefore, our forest only represents some of the derivations for a sentence given an SCFG rule table. The motivation of using such a forest is efficiency. However, since this space contains both "good" and "bad" translations, it still provides evidences for discriminative training.

First see the example in Figure 1. The derivation tree t_1 represented by solid hyperedges is a reference derivation. We can construct a non-reference derivation by making small change to t_1. By replacing the e_3 of t_1 with e_4, we obtain a non-reference deriva-

tion tree t_2. Considering the rules in each derivation, the difference between t_1 and t_2 lies in r_3 and r_4. Although r_3 has a same source side with r_4, it produces a different translation. While r_3 provides a swapping translation, r_4 generates a monotone translation. Thus, the derivation t_2 fails to move the subject "police" to the behind of verb "shot dead", resulting a wrong translation "the gunman was police shot dead". Given such derivations, we hope that the discriminative model is capable to explain why should use a reordering rule in this context.

Generally, our forest contains all the reference derivations $\mathcal{RT}$ for a sentence given a rule table, and some neighboring non-reference derivations $\mathcal{NT}$, which can be defined from $\mathcal{RT}$.

More formally, we call two hyperedges e^1 and e^2 are **competing hyperedges**, if their corresponding rules $r(e^1) = \langle \gamma^1, \alpha^1 \rangle$ and $r(e^2) = \langle \gamma^2, \alpha^2 \rangle$:

$$\gamma^1 = \gamma^2 \wedge \alpha^1 \neq \alpha^2 \quad (1)$$

This means they give different translations for a same source side. We use $C(e)$ to represent the set of competing hyperedges of e.

Two derivations $t^1 = \langle V^1, E^1 \rangle$ and $t^2 = \langle V^2, E^2 \rangle$ are **competing derivations** if there exists $e^1 \in E^1$ and $e^2 \in E^2$: [2]

$$V^1 = V^2 \wedge E^1 - e^1 = E^2 - e^2 \\ \wedge e^2 \in C(e^1) \quad (2)$$

In other words, derivations t^1 and t^2 only differ in e^1 and e^2, and these two hyperedges are competing hyperedges. We use $C(t)$ to represent the set of competing derivations of tree t, and $C(t,e)$ to represent the set of competing derivations of t if the competition occurs in hyperedge e in t.

Given a rule table, the set of reference derivations $\mathcal{RT}$ for a sentence is determined. Then, the set of non-reference derivations $\mathcal{NT}$ can be defined from $\mathcal{RT}$:

$$\cup_{t \in \mathcal{RT}} C(t) \quad (3)$$

Overall, our forest is the compact representation of $\mathcal{RT}$ and $\mathcal{NT}$.

[2]The definition of derivation tree is similar to forest, except that the tree contains exactly one tree while forest contains exponentially trees. In tree, the hyperedge degrades to edge.

Algorithm 1 Forest Generation

```
1:  procedure GENERATE(t)
2:      list ← t
3:      for v ∈ t in post order do
4:          e ← incoming edge of v
5:          append C(t, e) to list;
6:          for u ∈ child(v) from left to right do
7:              t_n ← OPERATE(t, u)
8:              if t_n ≠ t then
9:                  append t_n to list
10:                 for e' ∈ t_n ∧ e' ∉ t do
11:                     append C(t_n,e') to list
12:                 if SCORE(t) < SCORE(t_n) then
13:                     t ← t_n
14:     return t,list
```

4 Fast Generation

It is still slow to calculate the entire forest defined in Section 3, therefore we use a greedy decoding for fast generating a subset of the forest. Starting form a reference derivation, we try to slightly change the derivation into a new reference derivation. During this process, we collect the competing derivations of reference derivations. We describe the details of local operators for changing a derivation in section 4.1, and then introduce the creation of initial reference derivation with max score in Section 4.2.

For example, given derivation t_1, we delete the node $X_{0,1}$ and the related hyperedge e_1 and e_5. Fixing the other nodes and edges, we try to add a new edge e_2 to create a new reference translation. In this case, if rule r_2 really exists in our rule table, we get a new reference derivation t_3. After constructing t_3, we first collect the new tree and $C(t_3, e_2)$. Then, we will move to t_3, if the score of t_3 is higher than t_2. Notably, if r_2 does not exist in the rule table, we fail to create a new reference derivation. In such case, we keep the origin derivation unchanged.

Algorithm 1 shows the process of generation.[3] The input is a reference derivation t, and the output is a new derivation and the generated derivations.

[3]For simplicity, we list all the trees, and do not compress them into a forest in practice. It is straight to extent the algorithm to get a compact forest for those generated derivations. Actually, instead of storing the derivations, we call the generate function twice to calculate gradient of log-linear model.

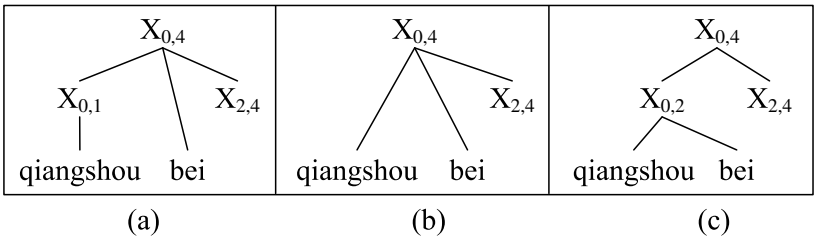


Figure 2: Lexicalize and generalize operators over t_1 (part) in Figure 1. Although here only shows the nodes, we also need to change relative edges actually. (1) Applying lexicalize operator on the non-terminal node $X_{0,1}$ in (a) results a new derivation shown in (b). (2) When visiting *bei* in (b), the generalize operator changes the derivation into (c).

The $list$ used for storing forest is initialized with the input tree (line 2). We visit the nodes in t in post-order (line 3). For each node v, we first append the competing derivations $C(t,e)$ to $list$, where e is incoming edge of v (lines 4-5). Then, we apply operators on the child nodes of v from left to right (lines 6-13). The operators returns a reference derivation t_n (line 7). If it is new (line 8), we collect both the t_n (line 9), and also the competing derivations $C(t_n, e')$ of the new derivation on those edges e' which only occur in the new derivation (lines 10-11). Finally, if the new derivation has a larger score, we will replace the origin derivation with new one (lines 12-13).

Although there is a two-level loop for visiting nodes (line 3 and 6), each node is visited only one time in the inner loops. Thus, the complexity is linear with the number of nodes $\#node$. Considering that the number of source word (also leaf node here) is less than the total number of nodes and is more than $\lceil(\#node+1)/2\rceil$, the time complexity of the process is also linear with the number of source word.

4.1 Lexicalize and Generalize

The function OPERATE in Algorithm 1 uses two operators to change a node: *lexicalize* and *generalize*. Figure 2 shows the effects of the two operators. The **lexicalize** operator works on nonterminal nodes. It moves away a nonterminal node and attaches the children of current node to its parent. In Figure 2(b), the node $X_{0,1}$ is deleted, requiring a more lexicalized rule to be applied to the parent node $X_{0,4}$ (one more terminal in the source side). We constrain the lexicalize operator to apply on pre-terminal nodes whose children are all terminal nodes. In contrast, the **generalize** operator works on terminal nodes and inserts a nonterminal node between current node and its parent node. This operator generalizes over the continuous terminal sibling nodes left to the current node (including the current node). Generalizing the node *bei* in Figure 2(b) results Figure 2(c). A new node $X_{0,2}$ is inserted as the parent of node *qiangshou* and node *bei*.

Notably, there are two steps when apply an operator. Suppose we want to lexicalize the node $X_{0,1}$ in t_1 of Figure 1, we first delete the node $X_{0,1}$ and related edge e_1 and e_5, then we try to add the new edge e_2. Since rule table is fixed, the second step is a process of decoding. Therefore, sometimes we may fail to create a new reference derivation (like r_2 may not exist in the rule table). In such case, we keep the origin derivation unchanged.

The changes made by the two operators are local. Considering the change of rules, the lexicalize operator deletes two rules and adds one new rule, while the generalize operator deletes one rule and adds two new rules. Such local changes provide us with a way to incrementally calculate the scores of new derivations. We use this method motivated by Gibbs Sampler (Blunsom et al., 2009) which has been used for efficiently learning rules. The different lies in that we use the operator for decoding where the rule table is fixing.

4.2 Initialize a Reference Derivation

The generation starts from an initial reference derivation with max score. This requires bi-parsing (Dyer, 2010) over the source sentence $\mathbf{f}$ and the reference translation $\mathbf{e}$. In practice, we may face three problems.

First is efficiency problem. Exhaustive search over the space under SCFG requires $\mathcal{O}(|f|^3|e|^3)$.

To parse quickly, we only visit the tight consistent (Zhang et al., 2008) bi-spans with the help of word alignment $\mathbf{a}$. Only visiting tight consistent spans greatly speeds up bi-parsing. Besides efficiency, adoption of this constraint receives support from the fact that heuristic SCFG rule extraction only extracts tight consistent initial phrases (Chiang, 2007).

Second is degenerate problem. If we only use the features as traditional SCFG systems, the bi-parsing may end with a derivation consists of some giant rules or rules with rare source/target sides, which is called degenerate solution (DeNero et al., 2006). That is because the translation rules with rare source/target sides always receive a very high translation probability. We add a prior score $log(\#rule)$ for each rule, where $\#rule$ is the number of occurrence of a rule, to reward frequent reusable rules and derivations with more rules.

Finally, we may fail to create reference derivations due to the limitation in rule extraction. We create minimum trees for $(\mathbf{f}, \mathbf{e}, \mathbf{a})$ using shift-reduce (Zhang et al., 2008). Some minimum rules in the trees may be illegal according to the definition of Chiang (2007). We also add these rules to the rule table, so as to make sure every sentence is reachable given the rule table. A source sentence is **reachable** given a rule table if reference derivations exists. We refer these rules as **added rules**. However, this may introduce rules with more than two variables and increase the complexity of bi-parsing. To tackle this problem, we initialize the chart with minimum parallel tree from the Zhang et al. (2008) algorithm, ensuring that the bi-parsing has at least one path to create a reference derivation. Then we only need to consider the traditional rules during bi-parsing.

5 Training

We use the forest to train a log-linear model with a latent variable as describe in Blunsom et al.(2008). The probability $p(\mathbf{e}|\mathbf{f})$ is the sum over all possible derivations:

$$p(\mathbf{e}|\mathbf{f}) = \sum_{\mathbf{t} \in \triangle(\mathbf{e},\mathbf{f})} p(\mathbf{t}, \mathbf{e}|\mathbf{f}) \quad (4)$$

where $\triangle(\mathbf{e}, \mathbf{f})$ is the set of all possible derivations that translate $\mathbf{f}$ into $\mathbf{e}$ and $\mathbf{t}$ is one such derivation.[4]

[4]Although the derivation is typically represent as $\mathbf{d}$, we denotes it by t since our paper use tree to represent derivation.

Algorithm 2 Training

procedure TRAIN($\mathcal{S}$)
  Training Data $\mathcal{S} = \{\mathbf{f}^n, \mathbf{e}^n, \mathbf{a}^n\}_{n=1}^N$
  Derivations $\mathcal{T} = \{\}_{n=1}^N$
  for $n = 1$ **to** N **do**
    $t^n \leftarrow$ INITIAL($\mathbf{f}^n, \mathbf{e}^n, \mathbf{a}^n$)
  $i \leftarrow 0$
  for $m = 0$ to M **do**
    for $n = 0$ to N **do**
      $\eta \leftarrow$ LEARNRATE(i)
      $(\Delta L(\mathbf{w}^i, t^n), t^n) \leftarrow$GENERATE($t^n$)
      $\mathbf{w}^i \leftarrow \mathbf{w}^i + \eta \times \Delta L(\mathbf{w}^i, t^n)$
      $i \leftarrow i + 1$
  return $\frac{\sum_{i=1}^{MN} \mathbf{w}^i}{MN}$

This model defines the conditional probability of a derivation $\mathbf{t}$ and the corresponding translation $\mathbf{e}$ given a source sentence $\mathbf{f}$ as:

$$p(\mathbf{t}, \mathbf{e}|\mathbf{f}) = \frac{\exp \sum_i \lambda_i h_i(\mathbf{t}, \mathbf{e}, \mathbf{f})}{Z(\mathbf{f})} \quad (5)$$

where the partition function is

$$Z(\mathbf{f}) = \sum_{\mathbf{e}} \sum_{\mathbf{t} \in \triangle(\mathbf{e},\mathbf{f})} \exp \sum_i \lambda_i h_i(\mathbf{t}, \mathbf{e}, \mathbf{f}) \quad (6)$$

The partition function is approximated by our forest, which is labeled as $\tilde{Z}(\mathbf{f})$, and the derivations that produce reference translation is approximated by reference derivations in $\tilde{Z}(\mathbf{f})$.

We estimate the parameters in log-linear model using maximum a posteriori (MAP) estimator. It maximizes the likelihood of the bilingual corpus $\mathcal{S} = \{\mathbf{f}^n, \mathbf{e}^n\}_{n=1}^N$, penalized using a gaussian prior (L2 norm) with the probability density function $p_0(\lambda_i) \propto \exp(-\lambda_i^2/2\sigma^2)$. We set σ^2 to 1.0 in our experiments. This results in the following gradient:

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = E_{p(\mathbf{t}|\mathbf{e},\mathbf{f})}[h_i] - E_{p(\mathbf{e}|\mathbf{f})}[h_i] - \frac{\lambda_i}{\sigma^2} \quad (7)$$

We use an online learning algorithm to train the parameters. We implement stochastic gradient descent (SGD) recommended by Bottou.[5] The dynamic learning rate we use is $\frac{N}{(i+i_0)}$, where N is the

[5]http://leon.bottou.org/projects/sgd

number of training example, i is the training iteration, and i_0 is a constant number used to get a initial learning rate, which is determined by calibration.

Algorithm 2 shows the entire process. We first create an initial reference derivation for every training examples using bi-parsing (lines 4-5), and then online learn the parameters using SGD (lines 6-12). We use the GENERATE function to calculate the gradient. In practice, instead of storing all the derivations in a list, we traverse the tree twice. The first time is calculating the partition function, and the second time calculates the gradient normalized by partition function. During training, we also change the derivations (line 10). When training is finished after M epochs, the algorithm returns an averaged weight vector (Collins, 2002) to avoid overfitting (line 13). We use a development set to select total epoch m, which is set as $M = 5$ in our experiments.

6 Experiments

Our method is able to train a large number of features on large data. We use a set of word context features motivated by word sense disambiguation (Chan et al., 2007) to test scalability. A word level context feature is a triple (f, e, f_{+1}), which counts the number of time that f is aligned to e and f_{+1} occurs to the right of f. Triple (f, e, f_{-1}) is similar except that f_{-1} locates to the left of f. We retain word alignment information in the extracted rules to exploit such features. To demonstrate the importance of scaling up the size of training data and the effect of our method, we compare three types of training configurations which differ in the size of features and data.

MERT. We use MERT (Och, 2003) to training 8 features on a small data. The 8 features is the same as Chiang (2007) including 4 rule scores (direct and reverse translation scores; direct and reverse lexical translation scores); 1 target side language model score; 3 penalties for word counts, extracted rules and glue rule. Actually, traditional pipeline often uses such configuration.

Perceptron. We also learn thousands of context word features together with the 8 traditional features on a small data using perceptron. Following (Chiang et al., 2009), we only use 100 most frequent words for word context feature. This setting use CKY decoder to generate n-best lists for training. The complexity of CKY decoding limits the training data into a small size. We fix the 8 traditional feature weights as MERT to get a comparable results as MERT.

	TRAIN	RTRAIN	DEV	TEST
#Sent.	519,359	186,810	878	3,789
#Word	8.6M	1.3M	23K	105K
Avg. Len.	16.5	7.3	26.4	28.0
Lon. Len.	99	95	77	116

Table 1: Corpus statistics of Chinese side, where Sent., Avg., Lon., and Len. are short for sentence, longest, average, and length respectively. RTRAIN denotes the reachable (given rule table without added rules) subset of TRAIN data.

Our Method. Finally, we use our method to train millions of features on large data. The use of large data promises us to use full vocabulary of training data for the context word features, which results millions of fully lexicalized context features. During decoding, when a context feature does not exit, we simply ignore it. The weights of 8 traditional features are fixed the same as MERT also. We fix these weights because the translation feature weights fluctuate intensely during online learning. The main reason may come from the degeneration solution mentioned in Section 4.2, where rare rules with very high translation probability are selected as the reference derivations. Another reason could be the fact that translation features are dense intensify the fluctuation. We leave learning without fixing the 8 feature weights to future work.

6.1 Data

We focus on the Chinese-to-English translation task in this paper. The bilingual corpus we use contains $519,359$ sentence pairs, with an average length of 16.5 in source side and 20.3 in target side, where $186,810$ sentence pairs (36%) are reachable (without added rules in Section 4.2). The monolingual data includes the Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We use the NIST evaluation sets of 2002 (MT02) as our development set, and sets of MT03/MT04/MT05 as test sets. Table 2 shows the statistics of all bilingual corpus.

We use GIZA++ (Och and Ney, 2003) to perform

System	#DATA	#FEAT	MT03	MT04	MT05	ALL
MERT	878	8	33.03	35.12	32.32	33.85
Perceptron	878	2.4K	32.89	34.88	32.55	33.76
Our Method	187K	2.0M	33.64	35.48	32.91*	34.41*
	519K	13.9M	34.19*	35.72*	33.09*	34.69*
Improvement over MERT			+1.16	+0.60	+0.77	+0.84

Table 2: Effect of our method comparing with MERT and perceptron in terms of BLEU. We also compare our fast generation method with different data (only reachable or full data). #Data is the size of data for training the feature weights. * means significantly (Koehn, 2004) better than *MERT* ($p < 0.01$).

word alignment in both directions, and grow-diag-final-and (Koehn et al., 2003) to generate symmetric word alignment. We extract SCFG rules as described in Chiang (2007) and also added rules (Section 4.2). Our algorithm runs on the entire training data, which requires to load all the rules into the memory. To fit within memory, we cut off those composed rules which only happen once in the training data. Here a composed rule is a rule that can be produced by any other extracted rules. A 4-grams language model is trained by the SRILM toolkit (Stolcke, 2002). Case-insensitive NIST BLEU4 (Papineni et al., 2002) is used to measure translation performance.

The training data comes from a subset of the LDC data including LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06. Since the rule table of the entire data is too large to be loaded to the memory (even drop one-count rules), we remove many sentence pairs to create a much smaller data yet having a comparable performance with the entire data. The intuition lies in that if most of the source words of a sentence need to be translated by the added rules, then the word alignment may be highly crossed and the sentence may be useless. We create minimum rules from a sentence pair, and count the number of source words in those minimum rules that are added rules. For example, suppose the result minimum rules of a sentence contain r_3 which is an added rule, then we count 1 time for the sentence. If the number of such source word is more than 10% of the total number, we will drop the sentence pair.

We compare the performances of MERT setting on three bilingual data: the entire data that contains 42.3M Chinese and 48.2M English words; 519K data that contains 8.6M Chinese and 10.6M English words; FBIS (LDC2003E14) parts that contains 6.9M Chinese and 9.1M English words. They produce 33.11/32.32/30.47 BLEU tested on MT05 respectively. The performance of 519K data is comparable with that of entire data, and much higher than that of FBIS data.

6.2 Result

Table 3 shows the performance of the three different training configurations. The training of MERT and perceptron run on MT02. For our method, we compare two different training sets: one is trained on all 519K sentence pairs, the other only uses 186K reachable sentences.

Although the perceptron system exploits 2.4K features, it fails to produce stable improvements over MERT. The reason may come from overfitting, since the training data for perceptron contains only 878 sentences. However, when use our method to learn the word context feature on the 519K data, we significantly improve the performance by 0.84 points on the entire test sets (ALL). The improvements range from 0.60 to 1.16 points on MT03-05. Because we use the full vocabulary, the number of features increased into 13.9 millions, which is impractical to be trained on the small development set. These results confirm the necessity of exploiting more features and learning the parameters on large data. Meanwhile, such results also demonstrate that we can benefits from the forest generated by our fast method instead of traditional CKY algorithm.

Not surprisingly, the improvements are smaller when only use 186K reachable sentences. Sometimes we even fail to gain significant improvement. This verifies our motivation to guarantee all sentence

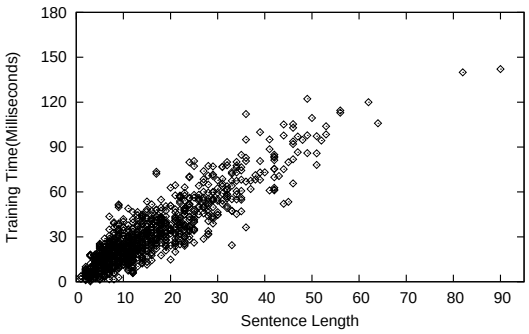


Figure 3: Plot of training times (including forest generation and SGD training) versus sentence length. We randomly select 1000 sentence from the 519K data for plotting.

are reachable, so as to use all training data.

6.3 Speed

How about the speed of our framework? Our method learns in 32 mlliseconds/sentence. Figure 3 shows training times (including forest generation and SGD training) versus sentence length. The plot confirms that our training algorithm scales linearly. If we use n-best lists which generated by CKY decoder as MERT, it takes about 3105 milliseconds/sentence for producing 100-best lists. Our method accelerates the speed about 97 times (even though we search twice to calculate the gradient). This shows the efficiency of our method.

The procedure of training includes two steps. (1) Bi-parsing to initialize a reference derivation with max score. (2) Training procedure which generates a set of derivations to calculate the gradient and update parameters. Step (1) only runs once. The average time of processing a sentence for each step is about 9.5 milliseconds and 30.2 milliseconds respectively.

For simplicity we do not compress the generated derivations into forests, therefore the size of resulting derivations is fairly small, which is about 265.8 for each sentence on average, where 6.1 of them are reference derivations. Furthermore, we use lexicalize operator more often than generalize operator (the ration between them is 1.5 to 1). Lexicalize operator is used more frequently mainly dues to that the reference derivations are initialized with reusable (thus small) rules.

7 Related Work

Minimum error rate training (Och, 2003) is perhaps the most popular discriminative training for SMT. However, it fails to scale to large number of features. Researchers have propose many learning algorithms to train many features: perceptron (Shen et al., 2004; Liang et al., 2006), minimum risk (Smith and Eisner, 2006; Li et al., 2009), MIRA (Watanabe et al., 2007; Chiang et al., 2009), gradient descent (Blunsom et al., 2008; Blunsom and Osborne, 2008). The complexity of n-best lists or packed forests generation hamper these algorithms to scale to a large amount of data.

For efficiency, we only use neighboring derivations for training. Such motivation is same as contrastive estimation (Smith and Eisner, 2005; Poon et al., 2009). The difference lies in that the previous work actually care about their latent variables (pos tags, segmentation, dependency trees, etc), while we are only interested in their marginal distribution. Furthermore, we focus on how to fast generate translation forest for training.

The local operators lexicalize/generalize are use for greedy decoding. The idea is related to "pegging" algorithm (Brown et al., 1993) and greedy decoding (Germann et al., 2001). Such types of local operators are also used in Gibbs sampler for synchronous grammar induction (Blunsom et al., 2009; Cohn and Blunsom, 2009).

8 Conclusion and Future Work

We have presented a fast generation algorithm for translation forest which contains both reference derivations and neighboring non-reference derivations for large-scale SMT discriminative training. We have achieved significantly improvement of 0.84 BLEU by incorporate 13.9M feature trained on 519K data in 0.03 second per sentence.

In this paper, we define the forest based on competing derivations which only differ in one rule. There may be better classes of forest that can produce a better performance. It's interesting to modify the definition of forest, and use more local operators to increase the size of forest. Furthermore, since the generation of forests is quite general, it's straight to

apply our forest on other learning algorithms. Finally, we hope to exploit more features such as reordering features and syntactic features so as to further improve the performance.

Acknowledgement

We would like to thank Yifan He, Xianhua Li, Daqi Zheng, and the anonymous reviewers for their insightful comments. The authors were supported by National Natural Science Foundation of China Contracts 60736014, 60873167, and 60903138.

References

Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proc. of EMNLP 2008*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL-08*.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A gibbs sampler for phrasal synchronous grammar induction. In *Proc. of ACL 2009*.

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

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proc. of ACL 2007*, pages 33–40.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proc. of NAACL 2009*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Trevor Cohn and Phil Blunsom. 2009. A Bayesian model of syntax-directed tree to string grammar induction. In *Proc. of EMNLP 2009*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP 2002*.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proc. of the HLT-NAACL 2006 Workshop on SMT*.

Chris Dyer. 2010. Two monolingual parses are better than one (synchronous parse). In *Proc. of NAACL 2010*.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proc. of ACL 2001*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL 2003*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP 2004*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proc. of EMNLP 2009*.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proc. of ACL 2009*.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of ACL 2006*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL 2008*.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL 2002*.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL 2002*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proc. of NAACL 2009*.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *Proc. of NAACL 2004*.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL 2005*.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proc. of COLING/ACL 2006*.

Andreas Stolcke. 2002. Srilm – an extensible language modeling toolkit. In *Proc. of ICSLP 2002*.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical mt. In *Proc. of ACL 2006*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proc. of EMNLP-CoNLL 2007*.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammar rules from word-level alignments in linear time. In *Proc. of Coling 2008*.

A Correction Model for Word Alignments

J. Scott McCarley, Abraham Ittycheriah, Salim Roukos, Bing Xiang, Jian-ming Xu
IBM T.J. Watson Research Center
1101 Kitchawan Road, Rt. 134
Yorktown Heights, NY 10598
{jsmc,abei,roukos,bxiang,jianxu}@us.ibm.com

Abstract

Models of word alignment built as sequences of links have limited expressive power, but are easy to decode. Word aligners that model the alignment matrix can express arbitrary alignments, but are difficult to decode. We propose an alignment matrix model as a correction algorithm to an underlying sequence-based aligner. Then a greedy decoding algorithm enables the full expressive power of the alignment matrix formulation. Improved alignment performance is shown for all nine language pairs tested. The improved alignments also improved translation quality from Chinese to English and English to Italian.

1 Introduction

Word-level alignments of parallel text are crucial for enabling machine learning algorithms to fully utilize parallel corpora as training data. Word alignments appear as hidden variables in IBM Models 1-5 (Brown et al., 1993) in order to bridge a gap between the sentence-level granularity that is explicit in the training data, and the implicit word-level correspondence that is needed to statistically model lexical ambiguity and word order rearrangements that are inherent in the translation process. Other notable applications of word alignments include cross-language projection of linguistic analyzers (such as POS taggers and named entity detectors,) a subject which continues to be of interest. (Yarowsky et al., 2001), (Benajiba and Zitouni, 2010)

The structure of the alignment model is tightly linked to the task of finding the optimal alignment. Many alignment models are factorized in order to use dynamic programming and beam search for efficient marginalization and search. Such a factorization encourages - but does not require - a sequential (often left-to-right) decoding order. If left-to-right decoding is adopted (and exact dynamic programming is intractable) important right context may exist beyond the search window. For example, the linkage of an English determiner may be considered before the linkage of a distant head noun.

An alignment model that jointly models all of the links in the entire sentence does not motivate a particular decoding order. It simply assigns comparable scores to the alignment of the entire sentence, and may be used to rescore the top-N hypotheses of another aligner, or to decide whether heuristic perturbations to the output of an existing aligner constitute an improvement. Both the training and decoding of full-sentence models have presented difficulties in the past, and approximations are necessary.

In this paper, we will show that by using an existing alignment as a starting point, we can make a significant improvement to the alignment by proposing a series of heuristic perturbations. In effect, we train a model to fix the errors of the existing aligner. From any initial alignment configuration, these perturbations define a multitude of paths to the reference (gold) alignment. Our model learns alignment moves that modify an initial alignment into the reference alignment. Furthermore, the resulting model assigns a score to the alignment and thus could be used in numerous rescoring algorithms, such as top-N rescorers.

In particular, we use the maximum entropy frame-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 889–898,
Edinburgh, Scotland, UK, July 27–31, 2011.

work to choose alignment moves. The model is symmetric: source and target languages are interchangeable. The alignment moves are sufficiently rich to reach arbitrary phrase to phrase alignments. Since most of the features in the model are not language-specific, we are able to test the correction model easily on nine language pairs; our corrections improved the alignment quality compared to the input alignments in all nine. We also tested the impact on translation and found a 0.48 BLEU improvement on Chinese to English and a 1.26 BLEU improvement on English to Italian translation.

2 Alignment sequence models

Sequence models are the traditional workhorse for word alignment, appearing, for instance, in IBM Models 1-5. This type of alignment model is not symmetric; interchanging source and target languages results in a different aligner. This parameterization does not allow a target word to be linked to more than one source word, so some phrasal alignments are simply not considered. Often the choice of directionality is motivated by this restriction, and the choice of tokenization style may be designed (Lee, 2004) to reduce this problem. Nevertheless, aligners that use this parameterization internally often incorporate various heuristics in order to augment their output with the disallowed alignments - for example, swapping source and target languages to obtain a second alignment (Koehn et al., 2007) with different limitations. Training both directions jointly (Liang et al., 2006) and using posterior probabilities during alignment prediction even allows the model to see limited right context. Another alignment combination strategy (Deng and Zhou, 2009) directly optimizes the size of the phrase table of a target MT system.

Generative models (such as Models 1-5, and the HMM model (Vogel et al., 1996)) motivate a narrative where alignments are selected left-to-right and target words are then generated conditioned upon the alignment and the source words. Generative models are typically trained unsupervised, from parallel corpora without manually annotated word-level alignments.

Discriminative models of alignment incorporate source and target words, as well as more linguistically motivated features into the prediction of alignment. These models are trained from annotated word alignments. Examples include the maximum entropy model of (Ittycheriah and Roukos, 2005) or the conditional random field jointly normalized over the entire sequence of alignments of (Blunsom and Cohn, 2006).

3 Joint Models

An alternate parameterization of alignment is the alignment matrix (Niehues and Vogel, 2008). For a source sentence F consisting of words $f_1 ... f_m$, and a target sentence $E = e_1 ... e_l$, the alignment matrix $A = \{\sigma_{ij}\}$ is an $l \times m$ matrix of binary variables. If $\sigma_{ij} = 1$, then e_i is said to be linked to f_j. If e_i is unlinked then $\sigma_{ij} = 0$ for all j. There is no constraint limiting the number of source tokens to which a target word is linked either; thus the binary matrix allows some alignments that cannot be modeled by the sequence parameterization. All 2^{lm} binary matrices are potentially allowed in alignment matrix models. For typical l and m, $2^{lm} \gg (m+1)^l$, the number of alignments described by a comparable sequence model. This parameterization is symmetric - if source and target are interchanged, then the alignment matrix is transposed.

A straightforward approach to the alignment matrix is to build a log linear model (Liu et al., 2005) for the probability of the alignment A. (We continue to refer to "source" and "target" words only for consistency of notation - alignment models such as this are indifferent to the actual direction of translation.) The log linear model for the alignment (Liu et al., 2005) is

$$p(A|E,F) = \frac{\exp\left(\sum_i \lambda_i \phi_i(A,E,F)\right)}{Z(E,F)} \quad (1)$$

where the partition function (normalization) is given by

$$Z(E,F) = \sum_A \exp\left(\sum_i \lambda_i \phi_i(A,E,F)\right). \quad (2)$$

Here the $\phi_i(A,E,F)$ are feature functions. The model is parameterized by a set of weights λ_i, one for each feature function. Feature functions are often binary, but are not required to be. Feature functions

may depend upon any number of components σ_{ij} of the alignment matrix A.

The sum over all alignments of a sentence pair (2^{lm} terms) in the partition function is computationally impractical except for very short sentences, and is rarely amenable to dynamic programming. Thus the partition function is replaced by an approximation. For example, the sum over all alignments may be restricted to a sum over the n-best list from other aligners (Liu et al., 2005). This approximation was found to be inconsistent for small n unless the merged results of several aligners were used. Alternately, loopy belief propagation techniques were used in (Niehues and Vogel, 2008). Loopy belief propagation is not guaranteed to converge, and feature design is influenced by consideration of the loops created by the features. Outside of the maximum entropy framework, similar models have been trained using maximum weighted bipartite graph matching (Taskar et al., 2005), averaged perceptron (Moore, 2005), (Moore et al., 2006), and transformation-based learning (Ayan et al., 2005).

4 Alignment Correction Model

In this section we describe a novel approach to word alignment, in which we train a log linear (maximum entropy) model of alignment by viewing it as correction model that fixes the errors of an existing aligner. We assume a priori that the aligner will start from an existing alignment of reasonable quality, and will attempt to apply a series of small changes to that alignment in order to correct it. The aligner naturally consists of a *move generator* and a *move selector*.

The move generator perturbs an existing alignment A in order to create a set of candidate alignments $\mathcal{M}_t(A)$, all of which are nearby to A in the space of alignments. We index the set of moves by the decoding step t to indicate that we generate entirely different (even non-overlapping) sets of moves at different steps t of the alignment prediction. Typically the moves affect linkages local to a particular word, e.g. the t'th source word.

The move selector then chooses one of the alignments $A_{t+1} \in \mathcal{M}_t(A_t)$, and proceeds iteratively: $A_{t+2} \in \mathcal{M}_{t+1}(A_{t+1})$, etc. until suitable termination criteria are reached. Pseudocode is depicted in Fig. (1.) In practice, one move for each source and target word is sufficient.

Input: sentence pair E_1 .. E_l, F_1 .. F_m
Input: alignment A
Output: improved alignment A_{final}
  for $t = 1 \rightarrow l$ **do**
    generate moves: $\mathcal{M}_t(A_t)$
    select move:
    $A_{t+1} \leftarrow \text{argmax}_{A \in \mathcal{M}_t(A_t)} p(A|A_t, E, F)$
  $A_{final} \leftarrow A_{l+1}$
  {repeat for source words}

Figure 1: pseudocode for alignment correction

4.1 Move generation

Many different types of alignment perturbations are possible. Here we restrict ourselves to a very simple move generator that changes the linkage of exactly one source word at a time, or exactly one target word at a time. Many of our corrections are similar to those of (Setiawan et al., 2010), although our motivation is perhaps closer to (Brown et al., 1993), who used similar perturbations to approximate intractable sums that arise when estimating the parameters of the generative models Models 3-5, and approach refined in (Och and Ney, 2003). We note that our corrections are designed to improve even a high-quality starting alignment; in contrast the model of (Fossum et al., 2008) considers deletion of links from an initial alignment (union of aligners) that is likely to overproduce links.

From the point of view of the alignment matrix, we consider changes to one row or one column (generically, one slice) of the alignment matrix. At each step t, the move set $\mathcal{M}_t(A_t)$ is formed by choosing a slice of the current alignment matrix A_t, and generating all possible alignments from a few families of moves. Then the move generator picks another slice and repeats. The $m + l$ slices are cycled in a fixed order: the first m slices correspond to source words (ordered according to a heuristic top-down traversal of the dependency parse tree if available), and the remaining l slices correspond to target words, similarly parse-ordered. For each slice we consider the following families of moves, illustrated by rows.

- add link to row i - for one j such that $\sigma_{ij} = 0$,

make $\sigma_{ij} = 1$ (shown here for row $i = 1$.)

	α	β	γ
a	○	○	○
b	○	●	○
c	○	○	○

$\Longrightarrow$

	α	β	γ
a	●	○	○
b	○	●	○
c	○	○	○

- remove one or more links from row i - for some j such that $\sigma_{ij} = 1$, make $\sigma_{ij} = 0$ (shown here for $i = 3$.)

	α	β	γ
a	●	○	○
b	○	●	○
c	○	○	●

$\Longrightarrow$

	α	β	γ
a	●	○	○
b	○	●	○
c	○	○	○

- move a link in row i - for one j and one j' such that $\sigma_{ij} = 1$ and $\sigma_{ij'} = 0$, make $\sigma_{ij} = 0$ and $\sigma_{ij'} = 1$ (shown here for $i = 1$.)

	α	β	γ
a	○	●	○
b	○	●	○
c	○	○	○

$\Longrightarrow$

	α	β	γ
a	●	○	○
b	○	●	○
c	○	○	○

- leave row i unchanged

Similar families of moves apply to column slices (source words.) In practice, perturbations are restricted by a window (typically ± 5 from existing links.) If the given source word is unlinked, we consider adding a link to each target word in a window (± 5 from nearby links.) The window size restrictions mean that some reference alignments are not reachable from the starting point. However, this is unlikely to limit performance - an oracle aligner achieves $97.6\% F$-measure on the Arabic-English training set.

4.2 Move selection

A log linear model for the selection of the candidate alignment at $t+1$ from the set of alignments $\mathcal{M}_t(A_t)$ generated by the move generator at step t takes the form:

$$p(A_{t+1}|E, F, \mathcal{M}_t(A_t)) = \frac{e^{\sum_i \lambda_i \phi_i(A_{t+1}, E, F)}}{Z(E, F, \mathcal{M}_t(A_t))} \quad (3)$$

where the partition function is now given by

$$Z(E, F, \mathcal{M}) = \sum_{A \in \mathcal{M}} e^{\sum_i \lambda_i \phi_i(A, E, F)} \quad (4)$$

and $A_{t+1} \in \mathcal{M}_t(A_t)$ is required for correct normalization. This equation is notationally very similar to equation (1), except that the predictions of the model are restricted to a small set of nearby alignments. For the move generator considered in this paper, the summation in Eq.(4) is similarly restricted, and hence training the model is tractable. The set of candidate alignments $\mathcal{M}_t(A_t)$ typically does not contain the reference (gold) alignment; we model the best alignment among a finite set of alternatives, rather than the correct alignment from among all possible alignments. This is a key difference between our model and (Liu et al., 2005).

Note that if we extended our definition of perturbation to the limiting case that the alignment set included all possible alignments then we would clearly recover the standard log linear model of alignment.

4.3 Training

Since the model is designed to predict perturbation to an alignment, it is trained from a collection of errorful alignments and corresponding reference sequences of aligner moves that reach the reference (gold) alignment. We construct a training set from a collection of sentence pairs and reference alignments for training $(A^{*n}, E^n, F^n)_{n=1}^N$, as well as collections of corresponding "first pass" alignments A_1^n produced by another aligner. For each n, we form a number of candidate alignment sets $\mathcal{M}_t(A_t^n)$, one for each source and target word. For training purposes, the true alignment from the set is taken to be the one identical with A^{*n} in the slice targeted by the move generator at the current step. (A small number of move sets do not have an exact match and are discarded.) Then we form an objective function from the log likelihood of reference alignment, smoothed with a gaussian prior

$$\mathcal{L} = \sum_n \mathcal{L}_n + \sum_i (\lambda_i / \gamma)^2 \quad (5)$$

where the likelihood of each training sample is

$$\mathcal{L}_n = \sum_{\alpha} \log p_1(A_n^0|E, F_n; \mathcal{M}(f_\alpha, A_n^0, E, F_n)) + \sum_{\beta} \log p_1(A_n^0|E, F_n; \mathcal{M}(e_\beta, A_n^0, E, F_n)) \quad (6)$$

The likelihood has a term for each sentence pair and for each decoder step. The model is trained by gradient ascent using the l-BFGS method (Liu and Nocedal, 1989), which has been successfully used for training log linear models (Blunsom and Cohn, 2006) in many natural language tasks, including alignment.

5 Features

A wide variety of features were used in the model. We group the features in three broad categories: link-based, geometrical, and parse-based.

Link-based features are those which decompose into a (linear) sum of alignment matrix elements σ_{ij}. An example link-based feature is one that fires if a source language noun is linked to a target language determiner. Note that this feature may fire more than once in a given sentence pair: as with most features in our model, it is an integer-valued feature that counts the number of times a structure appears in a sentence pair. These features do not capture any correlation between different σ_{ij}. Among the link-based features are those based on Model 1 translation matrix parameters $\tau(e_i|f_j)$ and $\tau(f_j|e_i)$. We bin the model 1 parameters, and form integer-valued features for each bin that count the number of links with $\tau_0 < \tau(e_i|f_j) < \tau_1$.

Geometrical features are those which capture correlation between different σ_{ij} based on adjacency or nearness. They capture the idea that nearby words in one language link to nearby words in the other language - the motivation of HMM-based models of alignment. An example is a feature that counts the number of times that the next word in the source language is linked to the next word in the target language:

$$\phi(A, E, F) = \sum_{ij} \sigma_{ij}\sigma_{i+1,j+1} \quad (7)$$

Parse-based features are those which capture correlation between different σ_{ij}, but use parsing to determine links which are correlated - for example, if a determiner links to the same word as its head noun. As an example, if e_i is the headword of $e_{i'}$, and f_j is the headword of $f_{j'}$, then

$$\phi(A, E, F) = \sum_{ij} \sigma_{ij}\sigma_{i'j'} \quad (8)$$

counts the number of times that a dependency relation in one language is preserved by alignment in the other language. This feature can also be decorated, either lexically, or with part-of-speech tags (as many features in all three categories are.)

5.1 Unsupervised Adaptation

We constructed a heuristic phrase dictionary for unsupervised adapatation. After aligning a large unannotated parallel corpus with our aligner, we enumerate fully lexicalized geometrical features that can be extracted from the resulting alignments - these are entries in a phrase dictionary. These features are tied, and treated as a single real-valued feature that fires during training and decoding phases if a set of hypothesized links matches the geometrical feature extracted from the unannotated data. The value of this real-valued feature is the log of the number of occurrences of the identical (lexicalized) geometrical feature in the aligned unannotated corpus.

6 Results

We design our experiments to validate that a correction model using simple features, mostly non-language-specific, can improve the alignment accuracy of a variety of existing aligners for a variety of language pairs; we do not attempt to exactly match features between comparison aligners - this is unlikely to lead to a robust correction model.

6.1 Arabic-English alignment results

We trained the Arabic-English alignment system on 5125 sentences from Arabic-English treebanks (LDC2008E61, LDC2008E22) that had been annotated for word alignment. Reference parses were used during the training. Results are measured on a 500 sentence test set, sampled from a wide variety of parallel corpora, including various genres. During alignment, only automatically-generated parses (based on the parser of (Ratnaparkhi, 1999)) were available. Alignments on

initial align	correction model	R (%)	P (%)	$F(\%)$	ΔF
GIZA++		76	76	76	
	corr(GIZA++)	86	94	90	14*
	corr(ME-seq)	88	92	90	14*
HMM		73	73	73	
	corr(HMM)	87	92	89	16*
	corr(ME-seq)	87	93	90	17*
ME-seq		82	84	83	
	corr(HMM)	88	92	90	7*
	corr(GIZA++)	87	94	91	8*
	corr(ME-seq)	89	94	**91**	8*

Table 1: Alignment accuracy for Arabic-English systems in percentage recall (R), precision(P), and F-measure. * denotes statistical significance (see text.)

lang	method	R (%)	P(%)	$F(\%)$	ΔF
ZH→EN	GIZA++	55	67	61	
	ME-seq	66	72	69	
	corr(ME-seq)	74	76	**75**	6*

Table 2: Alignment accuracy for Chinese(ZH)-English(EN) systems. * denotes statistical significance

lang	aligner	R(%)	P(%)	$F(\%)$	ΔF
IT→ EN	ME-seq	74	87	80	
	corr(ME-seq)	84	92	88	8*
EN→IT	ME-seq	75	86	80	
	corr(ME-seq)	84	92	88	8*
PT→EN	ME-seq	77	83	80	
	corr(ME-seq)	87	91	89	9†
EN→PT	ME-seq	79	87	83	
	corr(ME-seq)	88	90	89	6†
JA→EN	ME-seq	72	78	75	
	corr(ME-seq)	77	83	80	5*
RU→EN	ME-seq	81	85	83	
	corr(ME-seq)	82	92	87	4*
DE→EN	ME-seq	77	82	79	
	corr(ME-seq)	78	87	82	3*
ES→EN	ME-seq	93	86	90	
	corr(ME-seq)	92	88	90	0.6
FR→EN	ME-seq	89	91	90	
	corr(ME-seq)	88	92	90	0.1

Table 3: Alignment accuracy for additional languages. * denotes statistical significance; † statistical significance not available. IT=Italian, PT=Portuguese, JA=Japanese, RU=Russian, DE=German, ES=Spanish, FR=French

the training and test sets were decoded with three other aligners, so that the robustness of the correction model to different input alignments could be validated. The three aligners were GIZA++ (Och and Ney, 2003) (with the MOSES (Koehn et al., 2007) postprocessing option `-alignment grow-diag-final-and`) the posterior HMM aligner of (Ge, 2004), a maximum entropy sequential model (ME-seq) (Ittycheriah and Roukos, 2005). ME-seq is our primary point of comparison: it is discriminatively trained (on the same training data,) uses a rich set of features, and provides the best alignments of the three. Three correction models were trained: corr(GIZA++) is trained to correct the alignments produced by GIZA++, corr(HMM) is trained to correct the alignments produced by the HMM aligner, and corr(ME-seq) is trained to correct the alignments produced by the ME-seq model.

In Table (1) we show results for our system correcting each of the aligners as measured in the usual recall, precision, and F-measure.[1] The resulting improvements in F-measure of the alignments produced by our models over their corresponding baselines is statistically significant ($p < 10^{-4}$, indicated by a *.) Statistical significance is tested by a Monte Carlo bootstrap (Efron and Tibshirani, 1986) - sampling with replacement the difference in F-measure of the two system's alignments of the same sentence pair. Both recall and precision are improved, but the improvement in precision is somewhat larger. We also show cross-condition results in which a correction model trained to correct HMM alignments is applied to correct ME-seq alignments. These results show that our correction model is robust to different starting aligners.

6.2 Chinese-English alignment results

Table (2) presents results for Chinese-English word alignments. The training set for the corr(ME-seq) model consisted of approximately 8000 hand-aligned sentences sampled from LDC2006E93 and LDC2008E57. The model was trained to correct the output of the ME-seq aligner, and tested on the same condition. For this language pair, reference parses were not available in our training set, so automatically-generated parses were used for both training and test sets. Results are measured on a 512 sentence test set, sampled from a wide variety of parallel corpora of various genres. We compare performance with GIZA++, and with the ME-seq aligner. Again the resulting improvement over the ME-seq aligner is statistically significant. However, here the improvement in recall is somewhat larger than the improvement in precision.

6.3 Additional language pairs

Table (3) presents alignment results for seven other language pairs. Separate alignment corrector models were trained for both directions of Italian $\leftrightarrow$ English and Portuguese $\leftrightarrow$ English. The training and test data vary by language, and are sampled uniformly from a diverse set of corpora of various genres, including newswire, and technical manuals. Manual alignments for training and test data were annotated. We compare performance with the ME-seq aligner trained on the same training data. As with the Chinese results above, customization and feature development for the language pairs was minimal. In general, machine parses were always available for the English half of the pair. Machine parses were also available for French and Spanish. Machine part of speech tags were available for all language (although character-based heuristic was substituted for Japanese.) Large amounts (up to 10 million sentence pairs) of unaligned parallel text was available for model 1 type features. Our model obtained improved alignment F-measure in all language pairs, although the improvements were small for ES$\rightarrow$EN and FR$\rightarrow$EN, the language pairs for which the baseline accuracy was the highest.

6.4 Analysis

Some of the improvement can be attributed to "look-ahead" during the decoding. For example, the English word "the", which (during Arabic-English alignment) should often be aligned to the same Arabic words to which its headword is linked. The number of errors associated with "the" dropped from 383 (186 false alarms, 197 misses) in the ME-seq model to 137 (60 false alarms and 77 misses) in the current model.

In table 5, we show contributions to performance resulting from various classes of features. The

[1] We do not distinguish sure and possible links in our annotations - under this circumstance, alignment error rate(Och and Ney, 2003) is $1 - F$.

method	Zh-En correct	Zh-En miss	Zh-En fa	Ar-En correct	Ar-En miss	Ar-En fa
hmm				147	256	300
GIZA++	139	677	396	132	271	370
ME-seq	71	745	**133**	127	276	191
corr(ME-seq)	**358**	**458**	231	**264**	**139**	**114**

Table 4: Analysis of $2-1$ alignments errors (misses and false alarms) for Zh-En and Ar-En aligners

largest contribution is noted by removing features based on the Model 1 translation matrices. These features contain a wealth of lexical information learned from approximately 7×10^6 parallel sentences - information that cannot be learned from a relatively small amount of word-aligned training data. Geometrical features contribute more than parse-based features, but the contribution from parse-based features is important, and these are more difficult to incorporate into sequential models. We note that all of the comparison aligners had equivalent lexical information.

We show a small improvement from the unsupervised adaptation - learning phrases from the parallel corpus that are not captured by the lexical features based on model 1. The final row in the table shows the result of running the correction model on its own output. The improvement is not statistically significant, but it is important to note the performance is *stable* - a further indication that the model is robust to a wide variety of input alignments, and that our decoding scheme is a reasonable approach to finding the best alignment.

In table 4, we characterize the errors based on the fertility of the source and target words. We focus on the case that exactly one target word is linked to exactly two source words. These are the links that

feature	R(%)	P(%)	$F(\%)$	N_{exact}
base	89	94	91	136
base-M1	82	88	85	89
base-geometric	83	90	86	92
base-parse	87	93	90	116
base+un.adapt	89	94	92	141
+iter2	90	94	92	141

Table 5: Importance of feature classes - ablation experiments

alignment	corpus-level TER	corpus-level BLEU	p90 TER	p90 BLEU
ME-seq	56.06	32.65	64.20	21.31
corr(Me-seq)	56.25	33.10	63.47	22.02
both	56.07	33.13	63.41	22.14

Table 6: Translation results, Zh to En. BLEU=BLEUr4n4

alignment	TER	BLEUr1n4
ME-seq	35.02	69.94
corr(Me-seq)	33.10	71.20

Table 7: Translation results, En to It

are poorly suited for the HMM and ME-seq models used in this comparison because of the chosen directionality: the source (Arabic, Chinese) words are the states and the target (English) words are the observation. The HMM is able to produce these links only by the use of posterior probabilities, rather than viterbi decoding. The ME-seq model only produces these links because of language-specific post-processing. GIZA++ has an underlying sequential model, but uses both directionalities. The correction model improved performance across all three of these links structures. The single exception is that the number of $2-1$ false alarms increased (Zh-En alignments) but in this case, the first pass ME-seq alignment produced few false alarms because it simply proposed few links of this form. It is also notable that $1-2$ links are more numerous than $2-1$ links, in both language pairs. This is consequence of the choice of directionality and tokenization style.

6.5 Translation Impact

We tested the impact of improved alignments on the performance of a phrase-based translation system (Ittycheriah and Roukos, 2007) for three lan-

guage pairs. Our alignment did not improve the performance of a mature Arabic to English translation system, but two notable successes were obtained: Chinese to English, and English to Italian. It is well known that improved alignment performance does not always improve translation performance (Fraser and Marcu, 2007). A mature machine translation system may incorporate alignments obtained from multiple aligners, or from both directions of an asymmetric aligner. Furthermore, with large amounts of training data (the Gale Phase 4 Arabic English corpus consisting of 8×10^6 sentences,) a machine translation system is subject to a saturation effect: correcting an alignment may not yield a significant improvement because the the phrases learned from the correct alignment have already been acquired in other contexts.

For the Chinese to English translation system (table 6) the training corpus consisted of 11×10^6 sentence pairs, subsampled to 10^6. The test set was NIST MT08 Newswire, consisting of 691 sentences and 4 reference translations. Corpus-level performance (columns 2 and 3) improved when measured by BLEU, but not by TER. Performance on the most difficult sentences (near the 90th percentile, columns 4 and 5) improved on both BLEU and TER (Snover et al., 2006), and the improvement in BLEU was larger for the more difficult sentences than it was overall. Translation performance further improved, by a smaller amount, using both ME-seq and corr(ME-seq) alignments during the training.

The improved alignments impacted the translation performance of the English to Italian translation system (table 7) even more strongly. Here the training corpus consisted of 9.4×10^6 sentence pairs, subsampled to 387000 pairs. The test set consisted of 7899 sentences. Overall performance improved as measured by both TER and BLEU (1.26 points.)

7 Conclusions

A log linear model for the alignment matrix is used to guide systematic improvements to an existing aligner. Our system models arbitrary alignment matrices and allows features that incorporate such information as correlations based on parse trees in both languages. We train models to correct the errors of several existing aligners; we find the resulting models are robust to using different aligners as starting points. Improvements in alignment F-measure, often significant improvements, show that our model successfully corrects input alignments from existing models in all nine language pairs tested. The resulting Chinese-English and English-Italian word alignments also improved translation performance, especially on the English-Italian test, and notably on the particularly difficult subset of the Chinese sentences. Future work will assess its impact on translation for the other language pairs, as well as its impact on other tasks, such as named entity projection.

8 Acknowledgements

We would like to acknowledge the support of DARPA under Grant HR0011-08-C-0110 for funding part of this work. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

Necip Fazil Ayan, Bonnie J. Dorr, and Christof Monz. 2005. Alignment link projection using transformation-based learning. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 185–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yassine Benajiba and Imed Zitouni. 2010. Enhancing mention detection using projection via aligned corpora. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 993–1001. Association for Computational Linguistics.

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *In Proc. of ACL-2006*, pages 65–72.

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

Yonggang Deng and Bowen Zhou. 2009. Optimizing word alignment combination for phrase table training. In *Proceedings of the ACL-IJCNLP 2009 Conference*

Short Papers, ACLShort '09, pages 229–232, Stroudsburg, PA, USA. Association for Computational Linguistics.

B. Efron and R. Tibshirani. 1986. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1(1):pp. 54–75.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using syntax to improve word alignment precision for syntax-based machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 44–52. Association for Computational Linguistics.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Comput. Linguist.*, 33(3):293–303.

Niyu Ge. 2004. Improvement in word alignments. In *DARPA/TIDES MT workshop*.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *HLT-EMNLP*, pages 89–96.

Abraham Ittycheriah and Salim Roukos. 2007. Direct translation model 2. In *Human Language Technologies 2007: The Conference of the NA-ACL*, pages 57–64, Rochester, New York, April. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers on XX*, HLT-NAACL '04, pages 57–60. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

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

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 459–466. Association for Computational Linguistics.

Robert C. Moore, Wen-tau Yih, and Andreas Bode. 2006. Improved discriminative bilingual word alignment. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 513–520. Association for Computational Linguistics.

Robert C. Moore. 2005. A discriminative framework for bilingual word alignment. In *In Proceedings of HLT-EMNLP*, pages 81–88.

Jan Niehues and Stephan Vogel. 2008. Discriminative word alignment via alignment matrix modeling. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 18–25, Columbus, Ohio, June. Association for Computational Linguistics.

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

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Mach. Learn.*, 34:151–175, February.

Hendra Setiawan, Chris Dyer, and Philip Resnik. 2010. Discriminative word alignment with a function word reordering model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 534–544. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Ben Taskar, Simon Lacoste-julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *In Proceedings of HLT-EMNLP*, pages 73–80.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics*, pages 836–841.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, HLT '01, pages 1–8. Association for Computational Linguistics.

Heuristic Search for Non-Bottom-Up Tree Structure Prediction

Andrea Gesmundo
Department of Computer Science
University of Geneva
andrea.gesmundo@unige.ch

James Henderson
Department of Computer Science
University of Geneva
james.henderson@unige.ch

Abstract

State of the art Tree Structures Prediction techniques rely on bottom-up decoding. These approaches allow the use of context-free features and bottom-up features. We discuss the limitations of mainstream techniques in solving common Natural Language Processing tasks. Then we devise a new framework that goes beyond Bottom-up Decoding, and that allows a better integration of contextual features. Furthermore we design a system that addresses these issues and we test it on Hierarchical Machine Translation, a well known tree structure prediction problem. The structure of the proposed system allows the incorporation of non-bottom-up features and relies on a more sophisticated decoding approach. We show that the proposed approach can find better translations using a smaller portion of the search space.

1 Introduction

Tree Structure Prediction (TSP) techniques have become relevant in many Natural Language Processing (NLP) applications, such as Syntactic Parsing, Semantic Role Labeling and Hierarchical Machine Translation (HMT) (Chiang, 2007). HMT approaches have a higher complexity than Phrase-Based Machine Translation techniques, but exploit a more sophisticated reordering model, and can produce translations with higher Syntactic-Semantic quality.

TSP requires as inputs: a weighted grammar, $\mathbf{G}$, and a sequence of symbols or a set of sequences encoded as a Lattice (Chappelier et al., 1999). The input sequence is often a sentence for NLP applications. Tree structures generating the input sequence can be composed using rules, r, from the weighted grammar, $\mathbf{G}$. TSP techniques return as output a tree structure or a set of trees (forest) that generate the input string or lattice. The output forest can be represented compactly as a weighted hypergraph (Klein and Manning, 2001). TSP tasks require finding the tree, $\mathbf{t}$, with the highest score, or the best-k such trees. Mainstream TSP relies on Bottom-up Decoding (BD) techniques.

With this paper we propose a new framework as a generalization of the CKY-like Bottom-up approach. We also design and test an instantiation of this framework, empirically showing that wider contextual information leads to higher accuracy for TSP tasks that rely on non-local features, like HMT.

2 Beyond Bottom-up Decoding

TSP decoding requires scoring candidate trees, $\mathsf{cost}(\mathbf{t})$. Some TSP tasks require only local features. For these cases $\mathsf{cost}(\mathbf{t})$ depends only on the local score of the rules that compose $\mathbf{t}$:

$$\mathsf{cost}(\mathbf{t}) = \sum_{r_i \in \mathbf{t}} \mathsf{cost}(r_i) \qquad (1)$$

This is the case for Context Free Grammars. More complex tasks need non-local features. Those features can be represented by a non-local factor, $\mathsf{nonLocal}(\mathbf{t})$, into the overall $\mathbf{t}$ score:

$$\mathsf{cost}(\mathbf{t}) = \sum_{r_i \in \mathbf{t}} \mathsf{cost}(r_i) + \mathsf{nonLocal}(\mathbf{t}) \qquad (2)$$

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 899–908,
Edinburgh, Scotland, UK, July 27–31, 2011.

For example, in HMT the Language Model (LM) is a non-local fundamental feature that approximates the adequacy of the translation with the sum of log-probabilities of composing n-grams.

CKY-like BD approaches build candidate trees in a bottom-up fashion, allowing the use of Dynamic Programming techniques to simplify the search space by mering sub-trees with the same state, and also easing application of pruning techniques (such as Cube Pruning, e.g. Chiang (2007), Gesmundo (2010)). For clarity of presentation and following HMT practice, we will henceforth restrict our focus to binary grammars. Standard CKY works by building objects known as $items$ (Hopkins and Langmead, 2009). Each item, ι, corresponds to a candidate sub-tree. Items are built linking a rule instantiation, r, to two sub-items that represents left context, ι_1, and right context, ι_2; formally: $\iota \equiv \langle\ \iota_1 \succ r \prec \iota_2\ \rangle$. An item is a triple that contains a $span$, a $postcondition$ and a $carry$. The span contains the indexes of the starting and ending input words delimiting the continuous sequence covered by the sub-tree represented by the item. The postcondition is a string that represents r's head non-terminal label, telling us which rules may be applied. The carry, κ, stores extra information required to correctly score the non-local interactions of the item when it will be linked in a broader context (for HMT with LM the carry consists of boundary words that will form new n-grams).

Items, $\iota \equiv \langle \iota_1 \succ r \prec \iota_2 \rangle$, are scored according to the following formula:

$$\begin{aligned} \mathsf{cost}(\iota) &= \mathsf{cost}(r) + \mathsf{cost}(\iota_1) + \mathsf{cost}(\iota_2) \\ &+ \mathsf{interaction}(r, \kappa_1, \kappa_2) \end{aligned} \quad (3)$$

Where: $\mathsf{cost}(r)$ is the cost associated to the weighted rule r; $\mathsf{cost}(\iota_1)$ and $\mathsf{cost}(\iota_2)$ are the costs of the two sub-items computed recursively using formula (3); $\mathsf{interaction}(r, \kappa_1, \kappa_2)$ is the interaction cost between the rule instantiation and the two sub-items. In HMT the interaction cost includes the LM score of new n-grams generated by connecting the childrens' sub-spans with terminals of r. Notice that formula (3) is equal to formula (2) for items that cover the whole input sequence.

In many TSP applications, the search space is too large to allow an exhaustive search and therefore pruning techniques must be used. Pruning decisions are based on the score of partial derivations. It is not always possible to compute exactly non-local features while computing the score of partial derivations, since partial derivations miss part of the context. Formula (3) accounts for the interaction between r and sub-items ι_1 and ι_2, but it does not integrate the cost relative to the interaction between the item and the surrounding context. Therefore the item score computed in a bottom-up fashion is an approximation of the score the item has in a broader context. For example, in HMT the LM score for n-grams that partially overlap the item's span cannot be computed exactly since the surrounding words are not known.

Basing pruning decisions on approximated scores can introduce search errors. It is possible to reduce search errors using heuristics based on future cost estimation. In general the estimation of the interaction between ι and the surrounding context is function of the carry, κ. In HMT it is possible to estimate the cost of n-grams that partially overlap ι's span considering the boundary words. We can obtain the heuristic cost for an item, ι, adding to formula (3) the factor, $\mathsf{est}(\kappa)$, for the estimation of interaction with missing context:

$$\mathsf{heuristicCost}(\iota) = \mathsf{cost}(\iota) + \mathsf{est}(\kappa) \quad (4)$$

And use $\mathsf{heuristicCost}(\iota)$ to guide BD pruning decisions. Anyway, even if a good interaction estimation is available, in practice it is not possible to avoid search errors while pruning.

More sophisticated parsing models allow the use of non-bottom-up features within a BD framework. Caraballo and Charniak (1998) present best-first parsing with Figures of Merit that allows conditioning of the heuristic function on statistics of the input string. Corazza et al. (1994), and Klein and Manning (2003) propose an A* parsing algorithm that estimates the upper bound of the parse completion scores using contextual summary features. These models achieve time efficiency and state-of-the-art accuracy for PCFG parsing, but still use a BD framework that doesn't allow the application of a broader class of non-bottom-up contextual features.

In HMT, knowing the sentence-wide context in which a sub-phrase is translated is extremely important. It is obviously important for word choice: as

a simple example consider the translation of the frequent English word "get" into Chinese. The choice of the correct set of ideograms to translate "get" often requires being aware of the presence of particles that can be at any distance within the sentence. In a common English to Chinese dictionary we found 93 different sets of ideograms that could be translations of "get". Sentence-wide context is also important in the choice of word re-ordering: as an example consider the following translations from English to German:

1. EN : *I go home.*
   DE : *Ich gehe nach Hause.*

2. EN : *I say, that I go home.*
   DE : *Ich sage, dass ich nach Hause gehe.*

3. EN : *On Sunday I go home.*
   DE : *Am Sonntag gehe ich nach Hause.*

The English phrase "*I go home*" is translated in German using the same set of words but with different orderings. It is not possible to choose the correct ordering of the phrase without being aware of the context. Thus a bottom-up decoder without context needs to build all translations for "*I go home*", introducing the possibility of pruning errors.

Having shown the importance of contextual features, we define a framework that overcomes the limitations of bottom-up feature approximation.

3 Undirected-CKY Framework

Our aim is to propose a new Framework that overcomes BD limitations allowing a better integration of contextual features. The presented framework can be regarded as a generalization of CKY.

To introduce the new framework let us focus on a detail of CKY BD. The items are created and scored in topological order. The ordering constraint can be formally stated as: *an item covering the span* $[i, j]$ *must be processed after items covering sub spans* $[h, k] | h \geq i, k \leq j$. This ordering constraint implies that full yield information is available when an item is processed, but information about ancestors and siblings is missing. Therefore non-bottom-up context cannot be used because of the ordering constraint. Now let us investigate how the decoding algorithm could change if we remove the ordering constraint.

Removing the ordering constraint would lead to the occurrence of cases in which an item is processed before all child items have been processed. For example, we could imagine to create and score an item, ι, with postcondition X and span $[i, j]$, linking the rule instantiation $r : X \rightarrow AB$ with only the left sub-item, ι_A, while information for the right sub-item, ι_B is still missing. In this case, we can rely on local and partial contextual features to score ι. Afterwards, it is possible to process ι_B using the parent item, ι, as a source of additional information about the parent context and sibling ι_A yield. This approach can avoid search errors in cases where pruning at the parent level can be correctly done using only local and partial yield context, while pruning at the child level needs extra non-bottom-up context to make a better pruning decision. For example, consider the translation of the English sentence "*I run*" into French using the following synchronous grammar:

$$
\begin{aligned}
&r_1 : S \rightarrow X_{\boxed{1}}\, X_{\boxed{2}} \mid X_{\boxed{1}}\, X_{\boxed{2}} \\
&r_2 : X \rightarrow I \mid Je \\
&r_3 : X \rightarrow run \mid course \\
&r_4 : X \rightarrow run \mid courir \\
&r_5 : X \rightarrow run \mid cours \\
&r_6 : X \rightarrow run \mid courons \\
&\vdots
\end{aligned}
$$

Where: r_1 is a $Glue$ rule and boxed indexes describe the alignment; r_2 translates "I" in the corresponding French pronoun; r_3 translates "run" as a noun; remaining rules translate "run" as one of the possible conjugations of the verb "$courir$". Using only bottom-up features it is not possible to resolve the ambiguity of the word "run". If the beam is not big enough the correct translation could be pruned. Anyway a CKY decoder would give the highest score to the most frequent translation. Instead, if we follow a non bottom-up approach, as described in Figure 1, we can: 1) first translate "I"; 2) Then create an item using r_1 with missing right child; 3) finally choose the correct translation for "run" using r_1 to access a wider context. Notice that with this undirected approach it is possible to reach the correct translation using only beam size of

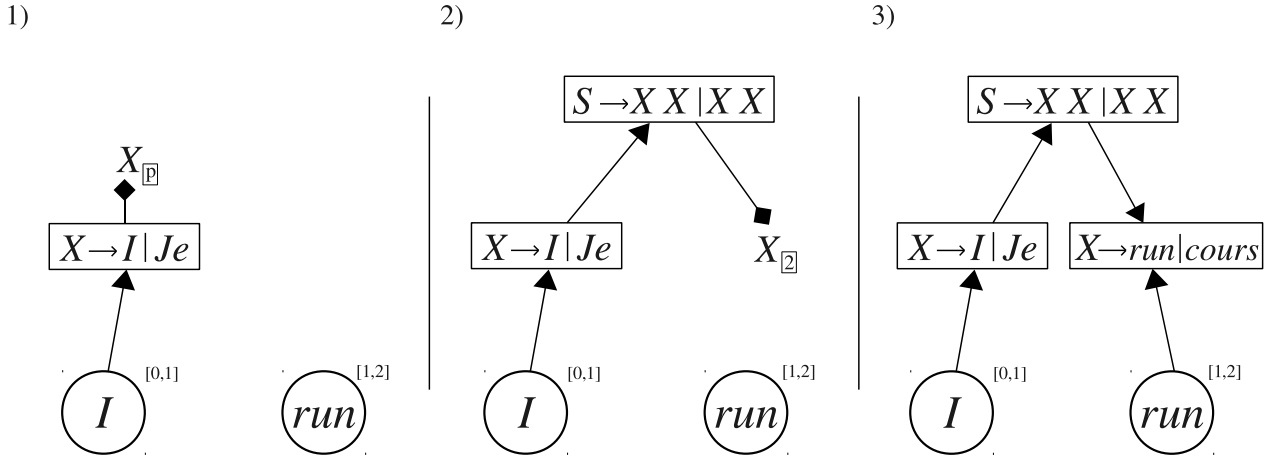


Figure 1: Example of undirected decoding for HMT. The arrows point to the direction in which information is propagated. Notice that the parent link at step 3 is fundamental to correctly disambiguate the translation for "run".

1 and the LM feature.

To formalize Undirected-CKY, we define a generalized item called $undirected\text{-}item$. Undirected-items, $\mathring{\boldsymbol{\imath}}$, are built linking rule instantiations with elements in $\mathcal{L} \equiv \{$left child, right child, parent$\}$; for example: $\mathring{\boldsymbol{\imath}} \equiv \langle \mathring{\boldsymbol{\imath}}_1 \gg r \dot{\vee} \mathring{\boldsymbol{\imath}}_p \rangle$, is built linking r with left child, $\mathring{\boldsymbol{\imath}}_1$, and parent, $\mathring{\boldsymbol{\imath}}_p$. We denote with $\mathcal{L}^+_{\mathring{\boldsymbol{\imath}}}$ the set of links for which the undirected-item, $\mathring{\boldsymbol{\imath}}$, has a connection, and with $\mathcal{L}^-_{\mathring{\boldsymbol{\imath}}}$ the set of missing links. An undirected-item is a triple that contains a span, a carry and an $undirected\text{-}postcondition$. The undirected-postcondition is a set of strings, one string for each of $\mathring{\boldsymbol{\imath}}$'s missing links, $l \in \mathcal{L}^-_{\mathring{\boldsymbol{\imath}}}$. Each string represents the non-terminal related to one of the missing links available for expansion. Bottom-up items can be considered specific cases of undirected-items having $\mathcal{L}^+ = \{$ left child, right child$\}$ and $\mathcal{L}^- = \{$parent$\}$. We can formally describe the steps of the example depicted in Figure 1 with:

$$1)\quad \frac{r_2 : X \rightarrow I|Je\ ,\ terminal : [0,1]}{\mathring{\boldsymbol{\imath}}_1 : [0,1,\{X_{\boxed{1}}\},\kappa_1]}$$

$$2)\quad \frac{r_1 : S \rightarrow X_{\boxed{1}}X_{\boxed{2}}|X_{\boxed{1}}X_{\boxed{2}}\ ,\ \mathring{\boldsymbol{\imath}}_1 : [0,1,\{X_{\boxed{1}}\},\kappa_1]}{\mathring{\boldsymbol{\imath}}_2 : [0,1,\{X_{\boxed{2}}\},\kappa_2)]}$$

$$3)\quad \frac{r_5 : X \rightarrow run|cours, \mathring{\boldsymbol{\imath}}_2 : [\cdots]\ ,\ terminal : [1,2]}{\mathring{\boldsymbol{\imath}}_3 : [0,2,\{\},\kappa_3]}$$

The scoring function for undirected-items can be obtained generalizing formula (3):

$$\begin{aligned} \mathsf{cost}(\mathring{\boldsymbol{\imath}}) &= \mathsf{cost}(r) \\ &+ \sum_{l \in \mathcal{L}^+} \mathsf{cost}(\mathring{\boldsymbol{\imath}}_l) \\ &+ \mathsf{interaction}(r, \mathcal{L}^+) \end{aligned} \tag{5}$$

In CKY, each span is processed separately in topological order, and the best-k items for each span are selected in sequence according to scoring function (4). In the proposed framework, the selection of undirected-items can be done in any order, for example: in a first step selecting an undirected-item for span s_1, then selecting an undirected-item for span s_2, and in a third step selecting a second undirected-item for s_1, and so on. As in agenda based parsing (Klein and Manning, 2001), all candidate undirected-items can be handled with an unique queue. Allowing the system to decide decoding order based on the candidates' scores, so that candidates with higher confidence can be selected earlier and used as context for candidates with lower confidence.

Having all candidates in the same queue introduces comparability issues. In CKY, candidates are comparable since each span is processed separately and each candidate is scored with the estimation of the yield score. Instead, in the proposed framework,

the unique queue contains candidates relative to different nodes and with different context scope. To ensure comparability, we can associate to candidate undirected-items a heuristic score of the full derivation:

$$\mathsf{heuristicCost}(\mathring{\boldsymbol{\imath}}) = \mathsf{cost}(\mathring{\boldsymbol{\imath}}) + \mathsf{est}(\mathring{\boldsymbol{\imath}}) \qquad (6)$$

Where $\mathsf{est}(\mathring{\boldsymbol{\imath}})$ estimates the cost of the missing branches of the derivation as a function of $\mathring{\boldsymbol{\imath}}'s$ partial structure and carry.

In this framework, the queue can be initialized with a candidate for each rule instantiation. These initializing candidates have no context information and can be scored using only local features. A generic decoding algorithm can loop selecting the candidate undirected-item with the highest score, $\mathring{\boldsymbol{\imath}}$, and then propagating its information to neighboring candidates, which can update using $\mathring{\boldsymbol{\imath}}$ as context. In this general framework the link to the parent node is not treated differently from links to children. While in CKY the information is always passed from children to parent, in Undirected-CKY the information can be propagated in any direction, and any decoding order is allowed.

We can summarize the steps done to generalize CKY into the proposed framework: 1) remove the node ordering constraint; 2) define the scoring of candidates with missing children or parent; 3) use a single candidate queue; 4) handle comparability of candidates from different nodes and/or with different context scope; 5) allow information propagation in any direction.

4 Undirected Decoding

In this section we propose Undirected Decoding (UD), an instantiation of the Undirected-CKY framework presented above. The generic framework introduces many new degrees of freedom that could lead to a higher complexity of the decoder. In our actual instantiation we apply constraints on the initialization step, on the propagation policy, and fix a search beam of k. These constraints allow the system to converge to a solution in practical time, allow the use of dynamic programming techniques to merge items with equivalent states, and gives us the possibility of using non-bottom-up features and testing their relevance.

Algorithm 1 Undirected Decoding

```
1:  function decoder (k) : out-forest
2:  Q ← LeafRules();
3:  while |Q| > 0 do
4:     ı̊ ← PopBest (Q);
5:     if CanPop(ı̊) then
6:        out-forest.Add(ı̊);
7:        if ı̊.HasChildrenLinks() then
8:           for all r ∈ HeadRules(ı̊) do
9:              Ĉ ← NewUndirectedItems(r, ı̊);
10:             for all ĉ ∈ Ĉ do
11:                if CanPop(ĉ) then
12:                   Q.Insert(ĉ);
13:                end if
14:             end for
15:          end for
16:       end if
17:    end if
18: end while
```

Algorithm 1 summarizes the UD approach. The beam size, k, is given as input. At $line$ 2 the queue of undirected-item candidates, $\mathbf{Q}$, is initialized with only leafs rules. At $line$ 3 the loop starts, it will terminate when $\mathbf{Q}$ is empty. At $line$ 4 the candidate with highest score, $\mathring{\boldsymbol{\imath}}$, is popped from $\mathbf{Q}$. $line$ 5 checks if $\mathring{\boldsymbol{\imath}}$ is within the beam width: if $\mathring{\boldsymbol{\imath}}$ has a span for which k candidates were already popped, then $\mathring{\boldsymbol{\imath}}$ is dropped and a new iteration is begun. Otherwise $\mathring{\boldsymbol{\imath}}$ is added to the out-forest at $line$ 6. From $line$ 7 to $line$ 10 the algorithm deals with the generation of new candidate undirected-items. $line$ 7 checks if $\mathring{\boldsymbol{\imath}}$ has both children links, if not a new decoding iteration is begun. $line$ 8 loops over the rule instantiations, r, that can use $\mathring{\boldsymbol{\imath}}$ as child. At $line$ 9, the set of new candidates, $\hat{\mathbf{C}}$, is built linking r with $\mathring{\boldsymbol{\imath}}$ and any context already available in the out-forest. Finally, between $line$ 10 and $line$ 12, each element $\hat{\mathbf{c}}$ in $\hat{\mathbf{C}}$ is inserted in $\mathbf{Q}$ after checking that $\hat{\mathbf{c}}$ is within the beam width: if $\hat{\mathbf{c}}$ has a span for which k candidates were already popped it doesn't make sense to insert it in $\mathbf{Q}$ since it will be surely discarded at $line$ 5.

In more detail, the function NewUndirectedItems$(r, \mathring{\boldsymbol{\imath}})$ at $line$ 9 creates new undirected-items linking r using: 1) $\mathring{\boldsymbol{\imath}}$ as child; 2) (optionally) as other child any other undirected-item that has already been inserted in the out-forest and

doesn't have a missing child and matches missing span coverage; 3) and using as parent context the best undirected-item with missing child link that has been incorporated in the out-forest and can expand the missing child link using r. In our current method, only the best possible parent context is used because it only provides context for ranking candidates, as discussed at the end of this section. In contrast, a different candidate is generated for each possible other child in 2), as well as for the case where no other child is included in the undirected-item.

We can make some general observations on the Undirected Decoding Algorithm. Notice that, the **if** statement at $line$ 7 and the way new undirected-items are created at $line$ 9, enforce that each undirected-item covers a contiguous span. An undirected-item that is missing a child link cannot be used as child context but can be used as parent context since it is added to the out-forest at $line$ 6 before the **if** statement at $line$ 7. Furthermore, the **if** statements at $line$ 5 and $line$ 11 check that no more than k candidates are selected for each span, but the algorithm does not require the the selection of exactly k candidates per span as in CKY.

The queue of candidates, $\mathbf{Q}$, is ordered according to the heuristic cost of formula (6). The score of the candidate partial structure is accounted for with factor $\mathsf{cost}(\mathring{\boldsymbol{\imath}})$ computed according to formula (5). The factor $\mathsf{est}(\mathring{\boldsymbol{\imath}})$ accounts for the estimation of the missing part of the derivation. We compute this factor with the following formula:

$$\mathsf{est}(\mathring{\boldsymbol{\imath}}) = \sum_{l \in \mathcal{L}^{-}_{\mathring{\boldsymbol{\imath}}}} \Big(\mathsf{localCost}(\mathring{\boldsymbol{\imath}}, l) + \mathsf{contextEst}(\mathring{\boldsymbol{\imath}}, l) \Big) \tag{7}$$

For each missing link, $l \in \mathcal{L}^{-}_{\mathring{\boldsymbol{\imath}}}$, we estimate the cost of the corresponding derivation branch with two factors: $\mathsf{localCost}(\mathring{\boldsymbol{\imath}}, l)$ that computes the context-free score of the branch with highest score that could be attached to l; and $\mathsf{contextEst}(\mathring{\boldsymbol{\imath}}, l)$ that estimates the contextual score of the branch and its interaction with $\mathring{\boldsymbol{\imath}}$. Because our model is implemented in the Forest Rescoring framework (e.g. Huang and Chiang (2007), Dyer et al. (2010), Li et al. (2009)), $\mathsf{localCost}(\mathring{\boldsymbol{\imath}}, l)$ can be efficiently computed exactly. In HMT it is possible to exhaustively represent and search the context-free-forest (ignoring the LM), which is done in the Forest Rescoring framework before our task of decoding with the LM. We exploit this context-free-forest to compute $\mathsf{localCost}(\mathring{\boldsymbol{\imath}}, l)$: for missing child links the $\mathsf{localCost}(\cdot)$ is the Inside score computed using the $(max,\ +)$ semiring (also known as the Viterbi score), and for missing parent links the $\mathsf{localCost}(\cdot)$ is the corresponding Outside score. The factor $\mathsf{contextEst}(\cdot)$ estimates the LM score of the words generated by the missing branch and their interaction with the span covered by $\mathring{\boldsymbol{\imath}}$. To compute the expected interaction cost we use the boundary words information contained in $\mathring{\boldsymbol{\imath}}$'s carry as done in BD. To estimate the LM cost of the missing branch we use an estimation function, conditioned on the missing span length, whose parameters are tuned on held-out data with gradient descent, using the search score as objective function.

To show that UD leads to better results than BD, the two algorithms are compared in the same search space. Therefore we ensure that candidates embedded in the UD out-forest would have the same score if they were scored from BD. We don't need to worry about differences derived from the missing context estimation factor, $\mathsf{est}(\cdot)$, since this factor is only considered while sorting the queue, $\mathbf{Q}$, according to the $\mathsf{heuristicCost}(\cdot)$. Also, we don't have to worry about candidates that are scored with no missing child and no parent link, because in that case scoring function (3) for BD is equivalent to scoring function (5) for UD. Instead, for candidates that are scored with parent link, we remove the parent link factor from the $\mathsf{cost}(\cdot)$ function when inserting the candidate into the out forest. And for the candidates that are scored with a missing child, we adjust the score once the link to the missing child is created in the out-forest. In this way UD and BD score the same derivation with the same score and can be regarded as two ways to explore the same search space.

5 Experiments

In this section we test the algorithm presented, and empirically show that it produces better translations searching a smaller portion of the search space.

We implemented UD on top of a widely-used HMT open-source system, cdec (Dyer et al., 2010). We compare with cdec Cube Pruning BD. The ex-

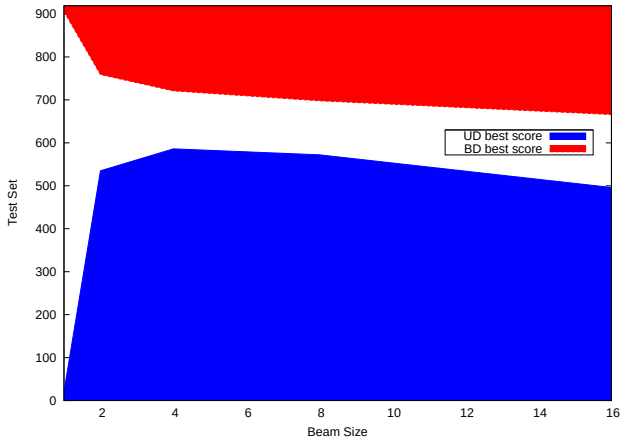


Figure 2: Comparison of the quality of the translations.

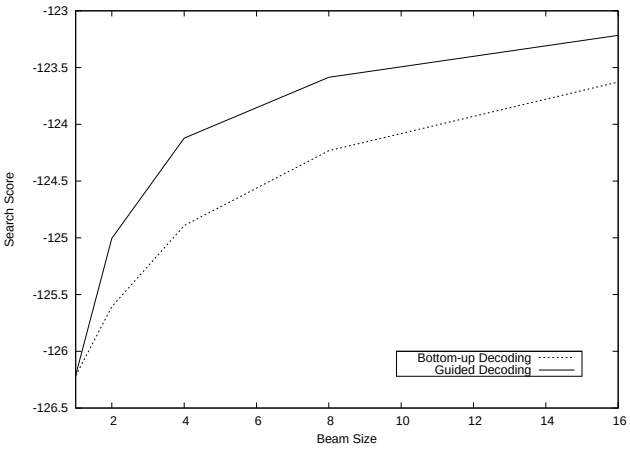


Figure 3: Search score evolution for BD and UD.

periments are executed on the NIST MT03 Chinese-English parallel corpus. The training corpus contains 239k sentence pairs with 6.9M Chinese words and 8.9M English words. We use a hierarchical phrase-based translation grammar extracted using a suffix array rule extractor (Lopez, 2007). The NIST-03 test set is used for decoding, it has 919 sentence pairs. The experiments can be reproduced on an average desktop computer. Since we compare two different decoding strategies that rely on the same training technique, the evaluation is primarily based on search errors rather than on BLEU. We compare the two systems on a variety of beam sizes between 1 and 16.

Figure 2 reports a comparison of the translation quality for the two systems in relation to the beam size. The blue area represents the portion of sentences for which UD found a better translation. The white area represents the portion of sentences for which the two systems found a translation with the same search score. With beam 1 the two systems obviously have a similar behavior, since both the systems stop investigating the candidates for a node after having selected the best candidate immediately available. For beams 2-4, UD has a clear advantage. In this range UD finds a better translation for two thirds of the sentences. With beam 4, we observe that UD is able to find a better translation for 63.76% of the sentences, instead BD is able to find a better translation for only 21.54% of the sentences. For searches that employ a beam bigger than 8, we notice that the UD advantage slightly decreases, and the number of sentences with equivalent translation slowly increases. We can understand this behavior considering that as the beam increases the two systems get closer to exhaustive search. Anyway with this experiment UD shows a consistent accuracy advantage over BD.

Figure 3 plots the search score variation for different beam sizes. We can see that UD search leads to an average search score that is consistently better than the one computed for BD. Undirected Decoding improves the average search score by 0.411 for beam 16. The search score is the logarithm of a probability. This variation corresponds to a relative gain of 50.83% in terms of probability. For beams greater than 8 we see that the two curves keep a monotonic ascendant behavior while converging to exhaustive search.

Figure 4 shows the BLEU score variation. Again we can see the consistent improvement of UD over BD. In the graph we report also the performance obtained using BD with beam 32. BD reaches BLEU score of 32.07 with beam 32 while UD reaches 32.38 with beam 16: UD reaches a clearly higher BLEU score using half the beam size. The difference is even more impressive if we consider that UD reaches a BLEU of 32.19 with beam 4.

In Figure 5 we plot the percentage reduction of the size of the hypergraphs generated by UD compared to those generated by BD. The size reduction grows quickly for both nodes and edges. This is due to the fact that BD, using Cube Pruning, must select k candidates for each node. Instead, UD is not obliged to

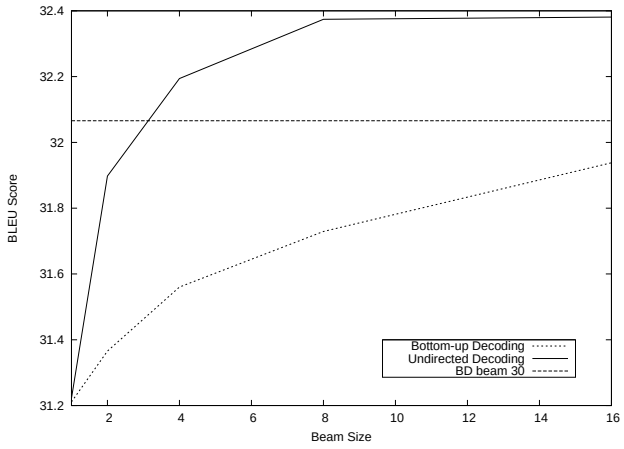


Figure 4: BLEU score evolution for BD and UD.

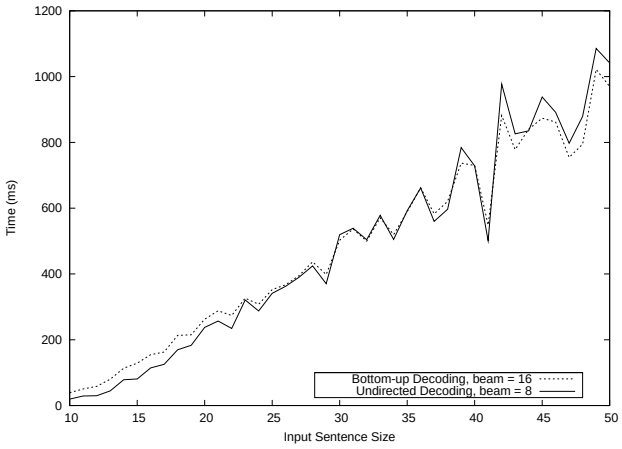


Figure 6: Time comparison between BD and UD.

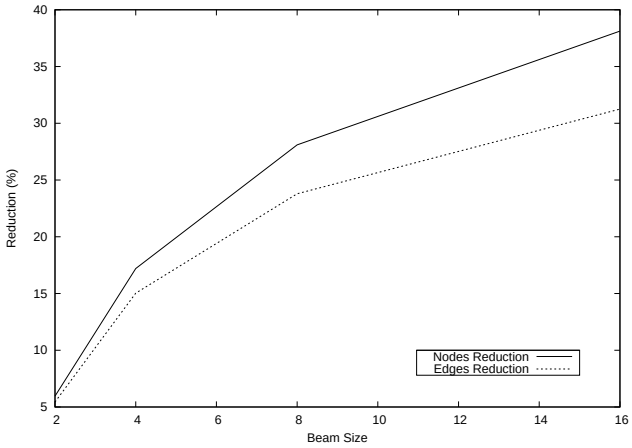


Figure 5: Percentage of reduction of the size of the hypergraph produced by UD.

select k candidates per $\mathbf{f}$-node. As we can see from Algorithm 1, the decoding loop terminates when the queue of candidates is empty, and the statements at $line\ 5$ and $line\ 11$ ensure that no more than k candidates are selected per $\mathbf{f}$-node, but nothing requires the selection of k elements, and some bad candidates may not be generated due to the sophisticated propagation strategy. The number of derivations that a hypergraph represents is exponential in the number of nodes and edges composing the structure. With beam 16, the hypergraphs produced by UD contain on average 4.6k fewer translations. Therefore UD is able to find better translations even if exploring a smaller portion of the search space.

Figure 6 reports the time comparison between BD and UD with respect to sentence length. The sentence length is measured with the number of ideogram groups appearing in the source Chinese sentences. We compare BD with beam of 16 and UD with beam of 8, so that we compare two systems with comparable search score. We can notice that for short sentences UD is faster, while for longer sentences UD becomes slower. To understand this result consider that for simple sentences UD can rely on the advantage of exploring a smaller search space. While, for longer sentences, the amount of candidates considered during decoding grows exponentially with the size of the sentence, and UD needs to maintain an unique queue whose size is not bounded by the beam size k, as for the queues used in BD's Cube Pruning. It may be possible to address this issue with more efficient handling of the queue.

In conclusion we can assert that, even if exploring a smaller portion of the search space, UD finds often a translation that is better than the one found with standard BD. UD's higher accuracy is due to its sophisticated search strategy that allows a more efficient integration of contextual features. This set of experiments show the validity of the UD approach and empirically confirm our intuition about the BD's inadequacy in solving tasks that rely on fundamental contextual features.

6 Future Work

In the proposed framework the link to the parent node is not treated differently from links to child nodes, the information in the hypergraph can be propagated in any direction. Then the Derivation

Hypergraph can be regarded as a non-directed graph. In this setting we could imagine applying message passing algorithms from graphical model theory (Koller and Friedman, 2010).

Furthermore, considering that the proposed framework lets the system decide the decoding order, we could design a system that explicitly learns to infer the decoding order at training time. Similar ideas have been successfully tried: Shen et al. (2010) and Gesmundo (2011) investigate the Guided Learning framework, that dynamically incorporates the tasks of learning the order of inference and training the local classifier.

7 Conclusion

With this paper we investigate the limitations of Bottom-up parsing techniques, widely used in Tree Structures Prediction, focusing on Hierarchical Machine Translation. We devise a framework that allows a better integration of non-bottom-up features. Compared to a state of the art HMT decoder the presented system produces higher quality translations searching a smaller portion of the search space, empirically showing that the bottom-up approximation of contextual features is a limitation for NLP tasks like HMT.

Acknowledgments

This work was partly funded by Swiss NSF grant CRSI22_127510 and European Community FP7 grant 216594 (CLASSiC, www.classic-project.org).

References

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing, Computational Linguistics, 24:275-298.

J. C. Chappelier and M. Rajman and R. Arages and A. Rozenknop. 1999. Lattice Parsing for Speech Recognition. In Proceedings of TALN 1999, Cargse, France.

David Chiang. 2007. Hierarchical phrase-based translation. Computational Linguistics, 33(2):201-228, 2007.

Anna Corazza, Renato De Mori, Roberto Gretter and Giorgio Satta. 1994. Optimal Probabilistic Evaluation Functions for Search Controlled by Stochastic Context-Free Grammars. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(10):1018-1027.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A Decoder, Alignment, and Learning framework for finite-state and context-free translation models. In Proceedings of the Conference of the Association of Computational Linguistics 2010, Uppsala, Sweden.

Andrea Gesmundo and James Henderson 2010. Faster Cube Pruning. Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT), Paris, France.

Andrea Gesmundo 2011. Bidirectional Sequence Classification for Tagging Tasks with Guided Learning. Proceedings of TALN 2011, Montpellier, France.

Mark Hopkins and Greg Langmead 2009. Cube pruning as heuristic search. Proceedings of the Conference on Empirical Methods in Natural Language Processing 2009, Singapore.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In Proceedings of the Conference of the Association of Computational Linguistics 2007, Prague, Czech Republic.

Dan Klein and Christopher D. Manning. 2001 Parsing and Hypergraphs, In Proceedings of the International Workshop on Parsing Technologies 2001, Beijing, China.

Dan Klein and Christopher D. Manning. 2003 A* Parsing: Fast Exact Viterbi Parse Selection, In Proceedings of the Conference of the North American Association for Computational Linguistics 2003, Edmonton, Canada.

Daphne Koller and Nir Friedman. 2010. Probabilistic Graphical Models: Principles and Techniques. The MIT Press, Cambridge, Massachusetts.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient Minimum Error Rate Training and Minimum Bayes-Risk decoding for translation hypergraphs and lattices, In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, Suntec, Singapore.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: An Open Source Toolkit for Parsing-based Machine Translation. In Proceedings of the Workshop on Statistical Machine Translation 2009, Athens, Greece.

Adam Lopez. 2007. Hierarchical Phrase-Based Translation with Suffix Arrays. In Proceedings of the Conference on Empirical Methods in Natural Language Processing 2007, Prague, Czech Republic.

Haitao Mi, Liang Huang and Qun Liu. 2008. Forest-Based Translation. In Proceedings of the Conference of the Association of Computational Linguistics 2008, Columbus, OH.

Libin Shen, Giorgio Satta and Aravind Joshi. 2007. Guided Learning for Bidirectional Sequence Classification. In Proceedings of the Conference of the Association of Computational Linguistics 2007, Prague, Czech Republic.

Andreas Stolcke. 2002. SRILM - An extensible language modeling toolkit. In Proceedings of the International Conference on Spoken Language Processing 2002, Denver, CO.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing, Proceedings of the Workshop on Statistical Machine Translation, New York City, New York.

Cache-based Document-level Statistical Machine Translation

Zhengxian Gong[1] **Min Zhang**[2] **Guodong Zhou**[1*]
[1] School of Computer Science and Technology
Soochow University, Suzhou, China 215006
[2] Institute for Infocomm Research, Singapore 138632
{zhxgong, gdzhou}@suda.edu.cn mzhang@i2r.a-star.edu.sg

Abstract

Statistical machine translation systems are usually trained on a large amount of bilingual sentence pairs and translate one sentence at a time, ignoring document-level information. In this paper, we propose a cache-based approach to document-level translation. Since caches mainly depend on relevant data to supervise subsequent decisions, it is critical to fill the caches with highly-relevant data of a reasonable size. In this paper, we present three kinds of caches to store relevant document-level information: 1) a dynamic cache, which stores bilingual phrase pairs from the best translation hypotheses of previous sentences in the test document; 2) a static cache, which stores relevant bilingual phrase pairs extracted from similar bilingual document pairs (i.e. source documents similar to the test document and their corresponding target documents) in the training parallel corpus; 3) a topic cache, which stores the target-side topic words related with the test document in the source-side. In particular, three new features are designed to explore various kinds of document-level information in above three kinds of caches. Evaluation shows the effectiveness of our cache-based approach to document-level translation with the performance improvement of 0.81 in BLUE score over Moses. Especially, detailed analysis and discussion are presented to give new insights to document-level translation.

1 Introduction

During last decade, tremendous work has been done to improve the quality of statistical machine translation (SMT) systems. However, there is still a huge performance gap between the state-of-the-art SMT systems and human translators. Bond (2002) suggested nine ways to improve machine translation by imitating the best practices of human translators (Nida, 1964), with parsing the entire document before translation as the first priority. However, most SMT systems still treat parallel corpora as a list of independent sentence-pairs and ignore document-level information.

Document-level information can and should be used to help document-level machine translation. At least, the topic of a document can help choose specific translation candidates, since when taken out of the context from their document, some words, phrases and even sentences may be rather ambiguous and thus difficult to understand. Another advantage of document-level machine translation is its ability in keeping a consistent translation.

However, document-level translation has drawn little attention from the SMT research community. The reasons are manifold. First of all, most of parallel corpora lack the annotation of document boundaries (Tam, 2007). Secondly, although it is easy to incorporate a new feature into the classical log-linear model (Och, 2003), it is difficult to capture document-level information and model it via some simple features. Thirdly, reference translations of a test document written by human translators tend to have flexible expressions in order to avoid producing monotonous texts. This makes the evaluation of document-level SMT systems extremely difficult.

Tiedemann (2010) showed that the repetition and consistency are very important when modeling natural language and translation. He proposed to employ cache-based language and translation models in a phrase-based SMT system for domain

* Corresponding author.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 909–919,
Edinburgh, Scotland, UK, July 27–31, 2011.

adaptation. Especially, the cache in the translation model dynamically grows up by adding bilingual phrase pairs from the best translation hypotheses of previous sentences. One problem with the dynamic cache is that those initial sentences in a test document may not benefit from the dynamic cache. Another problem is that the dynamic cache may be prone to noise and cause error propagation. This explains why the dynamic cache fails to much improve the performance.

This paper proposes a cache-based approach for document-level SMT using a static cache and a dynamic cache. While such a approach applies to both phrase-based and syntax-based SMT, this paper focuses on phrase-based SMT. In particular, the static cache is employed to store relevant bilingual phrase pairs extracted from similar bilingual document pairs (i.e. source documents similar to the test document and their target counterparts) in the training parallel corpus while the dynamic cache is employed to store bilingual phrase pairs from the best translation hypotheses of previous sentences in the test document. In this way, our cache-based approach can provide useful data at the beginning of the translation process via the static cache. As the translation process continues, the dynamic cache grows and contributes more and more to the translation of subsequent sentences.

Our motivation to employ similar bilingual document pairs in the training parallel corpus is simple: a human translator often collects similar bilingual document pairs to help translation. If there are translation pairs of sentences/phrases/words in similar bilingual document pairs, this makes the translation much easier. Given a test document, our approach imitates this procedure by first retrieving similar bilingual document pairs from the training parallel corpus, which has often been applied in IR-based adaptation of SMT systems (Zhao et al.2004; Hildebrand et al.2005; Lu et al.2007) and then extracting bilingual phrase pairs from similar bilingual document pairs to store them in a static cache.

However, such a cache-based approach may introduce many noisy/unnecessary bilingual phrase pairs in both the static and dynamic caches. In order to resolve this problem, this paper employs a topic model to weaken those noisy/unnecessary bilingual phrase pairs by recommending the decoder to choose most likely phrase pairs according to the topic words extracted from the target-side text of similar bilingual document pairs. Just like a human translator, even with a big bilingual dictionary, is often confused when he meets a source phrase which corresponds to several possible translations. In this case, some topic words can help reduce the perplexity. In this paper, the topic words are stored in a topic cache. In some sense, it has the similar effect of employing an adaptive language model with the advantage of avoiding the interpolation of a global language model with a specific domain language model.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents our cache-based approach to document-level SMT. Section 4 presents the experimental results. Session 5 gives new insights on cache-based document-level translation. Finally, we conclude this paper in Section 6.

2 Related work

There are only a few studies on document-level SMT. Representative work includes Zhao et al. (2006), Tam et al. (2007), Carpuat (2009).

Zhao et al. (2006) assumed that the parallel sentence pairs within a document pair constitute a mixture of hidden topics and each word pair follows a topic-specific bilingual translation model. It shows that the performance of word alignment can be improved with the help of document-level information, which indirectly improves the quality of SMT.

Tam et al. (2007) proposed a bilingual-LSA model on the basis of a parallel document corpus and built a topic-based language model for each language. By automatically building the correspondence between the source and target language models, this method can match the topic-based language model and improve the performance of SMT.

Carpuat (2009) revisited the "one sense per discourse" hypothesis of Gale et al. (1992) and gave a detailed comparison and analysis of the "one translation per discourse" hypothesis. However, she failed to propose an effective way to integrate document-level information into a SMT system. For example, she simply recommended some translation candidates to replace some target words in the post-process stage.

In principle, the cache-based approach can be well suited for document-level translation. Basical-

ly, the cache is analogous to "cache memory" in hardware terminology, which tracks short-term fluctuation (Iyer et al., 1999). As the cache changes with different documents, the document-level information should be capable of influencing SMT.

Previous cache-based approaches mainly point to cache-based language modeling (Kuhn and Mori, 1990), which uses a large global language model to mix with a small local model estimated from recent history data. However, applying such a language model in SMT is very difficult due to the risk of introducing extra noise (Raab, 2007).

For cache-based translation modeling, Nepveu et al. (2004) explored user-edited translations in the context of interactive machine translation. Tiedemann (2010) proposed to fill the cache with bilingual phrase pairs from the best translation hypotheses of previous sentences in the test document. Both Nepveu et al. (2004) and Tiedemann (2010) also explored traditional cache-based language models and found that a cache-based language model often contributes much more than a cache-based translation model.

3 Cache-based document-level SMT

Given a test document, our system works as follows:

1) clears the static, topic and dynamic caches when switching to a new test document d_x;
2) retrieves a set of most similar bilingual document pairs dd_s for d_x from the training parallel corpus using the cosine similarity with tf-idf weighting;
3) fills the static cache with bilingual phrase pairs extracted from dd_s;
4) fills the topic cache with topic words extracted from the target-side documents of dd_s;
5) for each sentence in the test document, translates it using cache-based SMT and continuously expands the dynamic cache with bilingual phrase pairs obtained from the best translation hypothesis of the previous sentences.

In this way, our cache-based approach can provide useful data at the beginning of the translation process via the static cache. As the translation process continues, the dynamic cache grows and contributes more and more to the translation of subsequent sentences. Besides, the possibility of choosing noisy/unnecessary bilingual phrase pairs in both the static and dynamic caches is wakened with the help of the topic words in the topic cache. In particular, only the most similar document pair is used to construct the static cache and the topic cache unless specified.

In this section, we first introduce the basic phrase-based SMT system and then present our cache-based approach to achieve document-level SMT with focus on constructing the caches (static, dynamic and topic) and designing their corresponding features.

3.1 Basic phrase-based SMT system

It is well known that the translation process of SMT can be modeled as obtaining the best translation **e** of the source sentence **f** by maximizing following posterior probability (Brown et al., 1993):

$$e_{best} = \arg\max_{e} P(e \mid f) = \arg\max_{e} P(f \mid e) P_{lm}(e) \quad (1)$$

where P(e|f) is a translation model and P_{lm} is a language model.

Our system adopted Moses (a state-of-art phrase-based SMT system) as a baseline, which follows Koehn et al. (2003) and mainly adopts six groups of popular features: 1) two phrase translation probabilities (two directions): $P_{phr}(e|f)$ and $P_{phr}(f|e)$; 2) two word translation probabilities (two directions) : $P_w(e|f)$ and $P_w(f|e)$; 3) one language model (target language): LM(e); 4) one phrase penalty (target language): PP(f); 5) one word penalty (target language):WP(e); 6) a lexicalized reordering model. Besides, the log-linear model as described in (Och and Ney, 2003) is employed to linearly interpolate these features for obtaining the best translation according to the formula (2):

$$e_{best} = \text{argmax} \{ \sum_{m=1}^{M} \lambda_m h_m(e, f) \} \quad (2)$$

where $h_m(e , f)$ is a feature function, and λ_m is the weight of $h_m(e , f)$ optimized by a discriminative training method on a held-out development data.

In principle, a phrase-based SMT system can provide the best phrase segmentation and alignment that cover a bilingual sentence pair. Here, a segmentation of a sentence into K phrases is defined as:

$$(f \sim e) \approx \sum_{i=1}^{K} (f_i, e_i, \sim) \quad (3)$$

where tuple (f_i, e_i) refers to a **phrase pair**, and ~ indicates corresponding alignment information.

3.2 Dynamic Cache

Our dynamic cache is mostly inspired by Tiedemann (2010), which adopts a dynamic cache to store relevant bilingual phrase pairs from the best translation hypotheses of previous sentences in the test document. In particular, a specific feature is incorporated S_{cache} to capture useful document-level information in the dynamic cache:

$$S_{cache}(e_c \mid f_c) = \frac{\sum_{i=1}^{K} I(< e_c, f_c >=< e_i, f_i >) \times e^{-\partial i}}{\sum_{i=1}^{K} I(f_c = f_i)} \quad (4)$$

where $e^{-\partial i}$ is a decay factor to avoid the dependence of the feature's contribution on the cache size. Given $<e_c, f_c>$ an existing phrase pair in the dynamic cache and $<e_i,f_i>$ a phrase pair in a new hypothesis, if ($e_i=e_c \wedge f_i=f_c$) is true (i.e. full matching), function I(.) returns 1 , otherwise 0.

One problem with the dynamic cache in Tiedemann (2010) is that it continuously updates the weight of a phrase pair in the dynamic cache. This may cause noticeable computational burden with the increasing number of phrase pairs in the dynamic cache. In addition, as a source phrase (f_c) may occur many times in the dynamic cache, the weights for related phrase pairs may degrade severely and thus his decoder needs a decay factor, which is difficult to optimize. Finally, Tiedemann (2010) only allowed full matching. This largely lowers down the probability of hitting the dynamic cache and thus much affects its effectiveness.

To overcome above problems, we only employ the bilingual phrase pairs in the dynamic cache to inform the decoder whether one bilingual phrase pair exists in the dynamic cache or not, which is slightly similar to (Nepveu et al, 2004) ,thus avoiding extra computational burden and the fine-tuning of the decay factor. In particular, following new feature is incorporated to better explore the dynamic cache:

$$F_d = \sum_{i=1}^{K} \mathrm{dpairmatch}(e_i, f_i) \quad (5)$$

where $\mathrm{dpairmatch}(e_i, f_i)$

$$= \begin{cases} 1 & (e_i = e_c \wedge f_i = f_c) \\ & \vee (\hat{e}_i = e_c \wedge \hat{f}_i = f_c \wedge \| e_c \| > 3) \\ & \vee (e_i = \hat{e}_c \wedge f_i = \hat{f}_c \wedge \| e_i \| > 3) \\ 0 & \text{other} \end{cases}$$

Here, F_d is called the dynamic cache feature. Assume (e_c,f_c) is a phrase pair in the dynamic cache and (e_i,f_i) is a phrase pair candidate for a new hypothesis. Besides full matching, we introduce a symbol of "^" for sub-phrase, such as $\hat{e}_i$ for a sub-phrase of e_i and $\hat{e}_c$ for a sub-phrase of e_c , to allow partial matching. Finally, F_d measures the overall value of a target candidate f_i by summing over the scores of K phrase pairs.

Obviously, F_d rewards both full matching and partial matching. In order to avoid too much noise, we put some constraints on the number of words in the target phrase of $<e_c,f_c>$ or $<e_i,f_i>$, such as $\| e_i \| > 3$, where " ||| " measures the number of non-blank characters in a phrase. For example, if phrase pair ", 减少||| and reduced" occurs in the cache, phrase pair ",|||and" is not rewarded because such shorter phrase pairs occur frequently and may largely degrade the effect of the cache. In accordance, the dynamic cache only contains phrase pairs whose target phrases contain 4 or more non-blank characters.

3.3 Static Cache

In Tiedemann (2010), initial sentences in a test document fail to benefit from the dynamic cache due to the lack of contents in the dynamic cache at the beginning of the translation process. To overcome this problem, a static cache is included to store relevant bilingual phrase pairs extracted from similar bilingual document pairs in the training parallel corpus. In particular, a static cache feature F_s is designed to capture useful information in the static cache in the same way as the dynamic cache feature, shown in Formula (5).

For this purpose, all the document pairs in the training parallel corpus are aligned at the phrase level using 2-fold cross-validation. That is, we adopt 50% of the training parallel corpus to train a model using Moses and apply the model to enforce phrase alignment of the remaining training data, and vice versa. Here, the enforcement is done by guaranteeing the occurrence of the target phrase candidate of a source phrase in the sentence pair. Besides, all the words pairs trained on the whole training parallel corpus are included in both folds to ensure at least one possible translation. Finally, the phrase pairs in the best translation hypothesis of a sentence pair is retrieved from the decoder. In this way, we can extract a set of phrase pairs for each bilingual document pairs.

Given a test document, we first find a set of similar source documents by computing the Cosine similarity using the TF-IDF weighting scheme and their corresponding target documents, from the training parallel corpus. Then, the phrase pairs ex-

tracted from these similar bilingual document pairs are collected into the static cache. To avoid noise, we filter out those phrase pairs which occur less than two times in the training parallel corpus.

出口 ||| exports	汇率 ||| exchange
放慢 ||| slowdown	活力 ||| vitality
股市 ||| stock market	加快 ||| speed up the
现行 ||| leading	经济学家 ||| economists
出口 增幅 ||| export growth	
多种 原因 ||| various reasons	
国家 著名 ||| a well-known international	
议会 委员会 ||| congressional committee	
不 乐观 的 预期 ||| pessimistic predictions	
保持 一定 的 增长 ||| maintain a certain growth	
美元 汇率 下跌 ||| a drop in the dollar exchange rate	

Table 1: Phrase pairs extracted from a document pair with an economic topic

Similar to the dynamic cache, we only consider those phrase pairs whose target phrases contain 4 or more non-blank characters to avoid noise. We do not deliberately remove long phrase pairs. It is possible to use these long phrase pairs if our test document is very similar to one training document pair. Table 1 shows some bilingual phrase pairs extracted from a document pair, which reports a piece of news about "impact on slowdown in US economic growth". Obviously, these phrase pairs are closely related to economics.

3.4 Topic Cache

Both the dynamic and static caches may still introduce noisy/unnecessary bilingual phrase pairs even with constraints on the length of phrases and their occurrence frequency in the training parallel corpus. In order to resolve this problem, this paper adopts a topic cache to store relevant topic words and employs a topic cache feature to weaken those noisy/unnecessary phrase pairs.

Given w_t is a topic word in the topic cache, the topic cache feature F_t is designed as follows:

$$F_t = \sum_{i=1}^{K} \text{topicexist}(e_i, f_i) \quad (6)$$

where $\text{topicexist}(e_i, f_i) = \begin{cases} 1 & (w_t \in e_i) \\ 0 & \text{other} \end{cases}$

Here, the target phrase which contains a topic word w_t will be rewarded. w_t is derived by a topic model, LDA (Latent Dirichlet Allocation). This is different from the previous work (Tam, 2007), which mainly interpolated a topic language model with a general language model and added additional two adaptive lexicon probabilities in his phrase table.

In principle, LDA is a probabilistic model of text data, which provides a generative analog of PLSA (Blei et al., 2003), and is primarily meant to reveal hidden topics in text documents. Like most of the text mining techniques, LDA assumes that documents are made up of words and the ordering of the words within a document is unimportant (i.e. the "bag-of-words" assumption).

Figure 1 shows the principle of LDA, where α is the parameter of the uniform Dirichlet prior on the per-document topic distributions, β is the parameter of the uniform Dirichlet prior on the per-topic word distribution, θ_i is the topic distribution for document i, z_{ij} is the topic for the jth word in document i, and w_{ij} is the specific word. Among all variables, w_{ij} is the only observable variable with all the other variables latent. In particular, K denotes the number of topics considered in the model and φ is a K*V (V is the dimension of the vocabulary) Markov matrix each line of which denotes the word distribution of a topic. The inner plate over z and w illustrates the repeated sampling of topics and words until N words have been generated for document d. The plate surrounding θ illustrates the sampling of a distribution over topics for each document d for a total of M documents. The plate surrounding φ illustrates the repeated sampling of word distributions for each topic z until K topics have been generated.

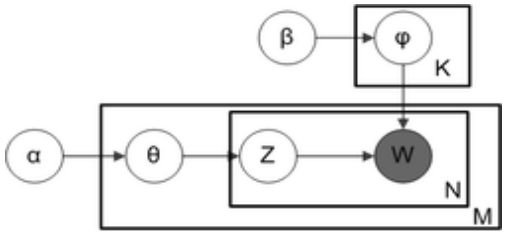


Figure 1: LDA

We use a LDA tool[1] to build a topic model using the target-side documents in the training parallel corpus. Using LDA, we can obtain the topic distribution of each word w, namely $p(z|w)$ for topic z ∈K. Moreover, using the obtained word topic distributions, we can infer the topic distribution of a new document, namely $p(z|d)$ for each topic z ∈K.

Given a test document, we first find the most similar source document from the training data in

[1] http://www.arbylon.net/projects/

the same way as done in the static cache. After that, we retrieve its corresponding target document. Then, the topic of the target document is determined by its major topic, with the maximum $p(z|d)$. Finally, we load some topic words corresponding to this topic z into the topic cache. In particular, our LDA model deploy the setting of K=15, α=0.5 and β=0.1. Besides, only top 1000 topic words are reserved for each topic. Table 2 shows top 10 topic words for five topics.

Topic 1	Topic 2	Topic 3	Topic4	Topic5
company	army	party	bush	election
corporation	armed	represents	united	olympic
limited	military	study	adminis-tration	games
manager	officers	theory	policy	votes
board	forces	leadership	president	bid
branch	units	political	clinton	gore
companies	troops	cadres	office	presi-dential
ltd	force	speech	secretary	party
business	soldiers	comrade	powell	won
personnel	police	central	relations	speech

Table 2: Topic words extracted from target-side documents

4 Experimentation

We have systematically evaluated our cache-based approach to document-level SMT on the Chinese-English translation task.

4.1 Experimental Setting

Here, we use SRI language modeling toolkit to train a trigram general language model on English newswire text, mostly from the Xinhua portion of the Gigaword corpus (2007) and performed word alignment on the training parallel corpus using GIZA++(Och and Ney,2000) in two directions. For evaluation, the NIST BLEU script (version 13) with the default setting is used to calculate the Bleu score (Papineni et al. 2002), which measures case-insensitive matching of n-grams with n up to 4. To see whether an improvement is statistically significant, we also conduct significance tests using the paired bootstrap approach (Koehn, 2004)[2]. In this paper, '***', '**', and '*' denote p-values less than or equal to 0.01, in-between (0.01, 0.05), and bigger than 0.05, which mean significantly better, moderately better and slightly better, respectively.

[2] http://www.ark.cs.cmu.edu/MT

In this paper, we use FBIS as the training data, the 2003 NIST MT evaluation test data as the development data, and the 2005 NIST MT test data as the test data. Table 3 shows the statistics of these data sets (with document boundaries annotated).

Corpus		Sentences	Documents
Role	Name		
Train	FBIS	239413	10353
Dev	NIST2003	919	100
Test	NIST2005	1082	100

Table 3: Corpus statistics

In particular, the sizes of the static, topic and dynamic caches are fine-tuned to 2000, 1000 and 5000 items, respectively. For the dynamic cache, we only keep those most recently-visited items, while for the static cache; we always keep the most frequently-occurring items.

4.2 Experimental Results

Table 4 shows the contribution of various caches in our cache-based document-level SMT system. The column of "BLEU_W" means the BLEU score computed over the whole test set and "BLEU_D" corresponds to the average BLEU score over separated documents.

System	BLEU on Dev(%)	BLEU on Test(%)		
		BLEU_W	NIST	BLEU_D
Moses	29.87	25.76	7.784	25.08
Fd	29.90	26.03 (*)	7.852	25.39
Fd+Fs	30.29	26.30 (**)	7.884	25.86
Fd+Ft	30.11	26.24 (**)	7.871	25.74
Fd+Fs+Ft	30.50	26.42 (***)	7.896	26.11
Fd+Fs+Ft with merging	-	26.57 (***)	7.901	26.32

Table 4: Contribution of various caches in our cache-based document-level SMT system. Note that significance tests are done against Moses.

Contribution of dynamical cache (Fd)

Table 4 shows that the dynamic cache slightly improves the performance by 0.27 (*) in BLEU_W. This is similar to Tiedemann (2010). However, detailed analysis indicates that the dynamic cache does have negative effect on about one third of documents, largely due to the instability of the dynamic cache at the beginning of translating a document. Figure 2 shows the distribution of the

BLEU_D difference of 100 test documents (sorted by BLEU_D). It shows that about 55% of test documents benefit from the dynamic cache.

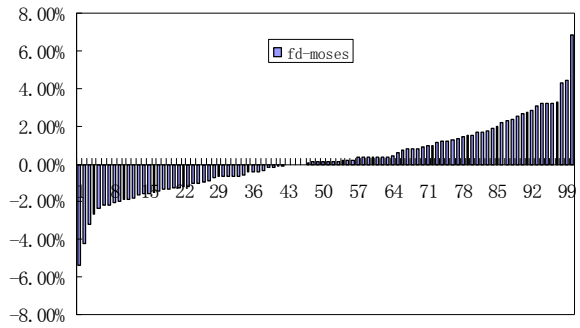


Figure 2: Contribution of employing the dynamic cache on different test documents

Contribution of static cache (Fs)

Table 4 shows that the combination of the static cache with the dynamic cache further improves the performance by 0.27(*) in BLEU_W. This suggests the effectiveness of the static cache in eliminating the instability of the dynamic cache when translating first few sentences of a test document. Together, the dynamic and static caches much improve the performance by 0.54 (**) in BLEU_W over Moses. Figure 3 shows the distribution of the BLEU_D difference of 100 test documents (sorted by BLEU_D), with more positive effect on those borderline documents, compared to Figure 2.

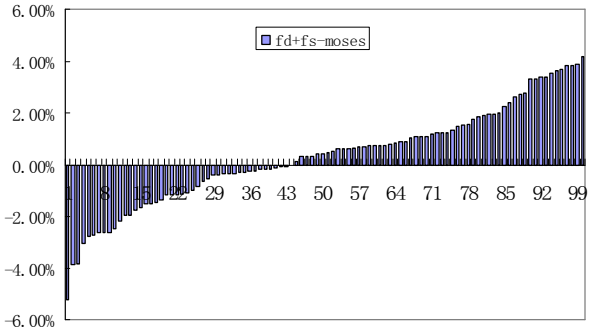


Figure 3: Contribution of combining the dynamic and static cache on different test documents

Contribution of topic cache (Ft)

Table 4 shows that the topic cache has comparable effect on improving the performance as the static cache when combined with the dynamic cache (0.48 vs. 0.54 in BLEU_W). Figure 4 shows the effectiveness of combining the dynamic and topic caches (sorted by BLEU_D).

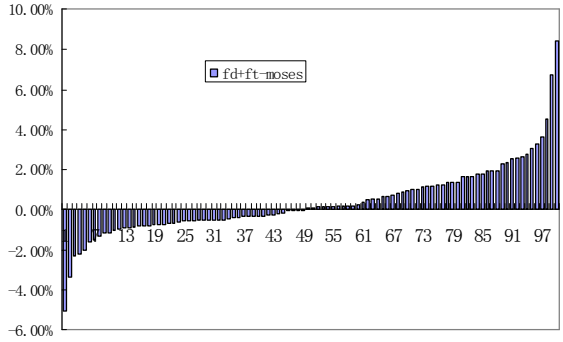


Figure 4: Contribution of combining the dynamic and topic caches

However, detailed analysis shows that the topic cache and the static cache are quite complementary by contributing on different test documents, largely due to that while the static cache tends to keep translation consistent, the topic cache plays like a document-specific language model. This is justified by Table 4 that the combination of the dynamic, static and topic caches significantly improve the performance by 0.66 (***) in BLEU_W, and by Figure 5 that about 75% of test documents benefit from the combination of the three caches (sorted by BLEU_D).

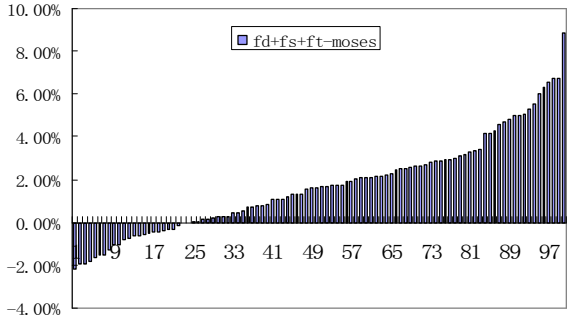


Figure 5: Contribution of combining the three caches

Contribution of merging phrase pairs of similar document pairs

Here, the number of similar documents we adopt is different from previous experiments. In the previous experiments, we only cache bilingual phrase pairs extracted from the most similar document. Here, we merge phrase pairs for several most similar documents (5 at most) which have the same topic.

Table 4 shows that employing this trick can further improve the performance by 0.15 in BLEU_W. As a result, the cache-based approach significantly improve the performance by 0.81 (***) in BLEU_W over Moses.

5 Discussion

In this section, we explore in more depth why the static cache can help the dynamic cache, some constrained factors which impact the effectiveness of our cache-based approach.

Effectiveness of the static cache

We investigate why the static cache affects the performance. Basically, it is difficult for the dynamic cache to capture such similar information in the static cache.

In principle, the static cache can both influence the initial and subsequent sentences; however subsequent ones can be affected by multiple caches. In order to give an insight of the static cache, we evaluate its effectiveness on the first sentence for each test document. Figure 6 shows the contribution of the static cache on translating these first sentences (y-axis shows BLEU value of the first sentence for each test document). It notes that the most BLEU scores of them are zeros because of the length limitation of first sentences.

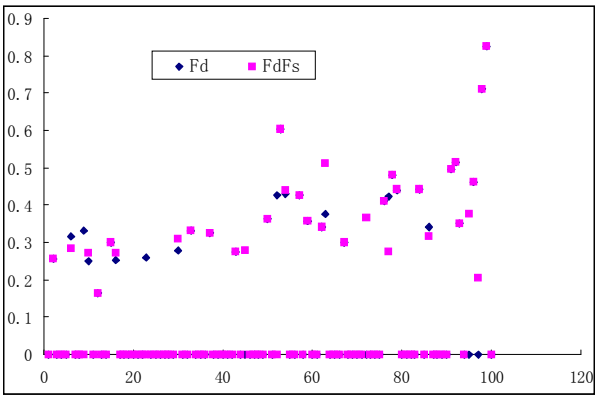


Figure 6: Contribution of the static cache on the first sentence of each test document (i.e. with empty dynamic cache)

Furthermore, we count the hit (matching) frequency of the static cache for each test documents. Since we use 1 or 0 for the static cache feature, it is easy to retrieve its effect for each test document. Our statistics shows that the hit frequency on static cache fluctuates between 5 and 18 for each test document. Without the static cache, the hit frequency of the dynamic cache is 504 on whole test sets, this figure increases to 685 with the static cache. This means that the static cache significantly enlarges the effectiveness of the dynamic cache by including more relevant phrase pairs to the dynamic cache, largely due to the positive impact of the static cache on the initial sentences of each test document.

Size of topic cache

Table 5 shows the impact of the topic cache when the number of the retained topic words for each topic increases from 500 to 2000. It shows that too more topic words actually harm the performance, due to the increase of noise. 1000 topic words seem a lot largely due to that we didn't do stemming for our topic modeling since we hope to introduce some tense information of them in the future.

Number of topic words	BLEU_W
500	26.27
700	26.31
1000	26.42
1500	26.23
2000	26.19

Table 5: Impact of the topic cache size

Influenced translations

In order to explore how our cache-based system impacts on translation results, we manually inspected 5 documents respectively which is improved or degraded in translation quality compared to the baseline Moses output. Those documents have 107 sentences in sum.

The good effectiveness of each kind of cache can be observed by the example 1 and 2 showed in Table 6. Both the example 1 and 2 come from the same document whose "BLEU_D" score exceeds Moses with 8.4 point. The example 1 benefits from the topic cache which contains the item of "action". The example 2 benefits from the static cache which contains a phrase pair of "承诺||| promised to" while Moses use "commitment" for "承诺" , which may be the reason for missing the part of "prime minister" in Moses output. Furthermore, due to the phrase pair of "停火 协议||| the ceasefire agreement" existing in our static cache, our decoder keeps using "ceasefire" to translate "停火" in the whole document while Moses randomly use "ceasefire" or "cease-fire" for this translation.

1	官员 预测 “ 准备 工作 将 会 进行 到 七月 , 然后 再 展开 政治 动作 ”
	Moses: official forecasts said that preparatory work will be carried out in july and then launched a political **maneuver** .
	Ours: official forecasts said that preparatory work will be carried out in july , then began a political **action** .
	Reference: officials expected that "preparations would take place until July, after which political action will begin".
2	关于 这 一 点 , 中东新闻社 说 , 以色列 总理 夏隆 承诺 “ 只 要 巴勒斯坦 当局 尊重 停火 协议 , 控制 好 它们 的 地方 , 以色列 将 会 停止 对 巴勒斯坦 人 的 军事 行动 ” 。
	Moses: on this point , said that israeli **commitment** to the palestinian authorities to respect the **cease-fire** agreement , where they are well under control , israel will stop its military actions against palestinians .
	Ours: on this point , said that israeli **prime minister promised to** respect the **ceasefire** agreement , the palestinian authorities to properly control their areas and israel will stop its military actions against palestinians .
	Reference:For this point , MENA said Israeli Prime Minister Sharon has promised to " stop Israeli military operations against the Palestinians insofar as they continue to respect the ceasefire deal and control their territory . "
3	17 日 晚 , 近 3000 多 名 市民 在 市中心 的 武器 广场 观看 了 由 市政府 举办 的 精彩纷呈 的 歌舞 晚会 , 五颜六色 的 灯光 装扮 着 广场 周围 的 古老 建筑 , 著名 歌舞 艺术家 们 表演 了 不同 地区 的 民族 歌舞 。
	Moses: on the evening , nearly 3,000 residents in the downtown square of the weapons held by the municipal government , watched a song and dance soiree , having colorful lighting disguise of ancient buildings around the square , singing and dancing famous artists staged different regions of ethnic song and dance .
	Ours: later on, nearly 3,000 residents in the downtown square to watch the government of having a song and dance performances were held under the disguise of colorful lighting around the square , a famous ancient buildings and local artists of different ethnic song and dance .
	Reference: On the night of the 17th , nearly 3,000 residents watched a wonderful gala of songs and dances , organized by the municipal government , at Plaza da Armas . Colorful lights lighted up ancient architecture around the plaza . Famous artists including singers and dancers staged performances of national songs and dances of different regions .
4	利马 的 城市 面积 已 从 建城 之 初 的 2.14 平方公里 发展 到 2600 多 平方公里 , 而 人口 也 增加 到 800 万 左右 , 约 占 全国 总人口 的 31% 。
	Moses: at lima 's urban area from the beginning of 2600 **square** to 2.14 million **square kilometers** , while the population has increased to 8 percent of the country 's total , about 31% .
	Ours: lima , the urban area from the beginning of 2600 **square kilometers** to 2.14 million **square kilometers** , but also increased to about 8 million population , the country 's total population of about 31% .
	Reference: The area of Lima city has expanded to more than 2,600 square kilometers from the original 2.14 square kilometers when the city was founded , while the population has increased to around 8 million , roughly accounting for 31% of the nation's total .

Table 6: Positive and negative examples

The example 3 and 4 also come from the same document however whose performance degrades with 2.17 point. We don't think the translation quality for example 4 in our system is worse than Moses. However, the translation quality for example 3 in our system is very bad and especially showed on "re-ordering". We found this sentence did not match any item in our static cache and topic cache. Although this phenomenon also happens in other documents, but this is the most typical negative example among these documents.

Document-specific characteristics

It seems that using the same weight for the whole test sets (all documents) is not very reasonable. Actually, if we can determine those negative documents which are not suitable for the cache-based approach, our cache-based approach may gain much improvement. Tiedemann (2010) explored the correlation to document length, baseline performance and source document repetition. However, it seems that there are no obvious rules to filter out those negative documents. Besides, there may be two more document-specific factors: repetition of the reference text and document style.

Tiedemann (2010) only considered the repetition of the test text in the source side. Since BLEU score is computed against the reference text, the repetition in the reference text may greatly influence the performance of our cache-based approach to document-level SMT. As for document style, it is quite possible that a document may contain several topics. Therefore, it may be useful to track such change over topics and refresh various caches when there is a topic change. We will leave the above issues to the future work.

6 Conclusion

We have shown that our cache-based approach significantly improves the performance with the help of various caches, such as the dynamic, static and topic caches, although the cache-based ap-

proach may introduce some negative impact on BLEU scores for certain documents.

In the future, we will further explore how to reflect document divergence during training and dynamically adjust cache weights according to different documents.

There are many useful components in training documents, such as named entity, event and coreference. In this experiment, we only adopt the flat data in our cache. However, the structured data may improve the correctness of matching and thus effectively avoid noise. We will explore more effective ways to pick up various kinds of useful information from the training parallel corpus to expand our cache-based approach. Besides, we will resort to comparable corpora to enlarge our cache-based approach to document-level SMT.

Acknowledgments

This research was supported by Projects 90920004, 60873150, and 61003155 under the National Natural Science Foundation of China, Project 20093201110006 under the Specialized Research Fund for the Doctoral Program of Higher Education of China.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. Journal of machine Learning Research 3, pages 993–1022.

Francis Bond. 2002. Toward a Science of Machine Translation. Asian Association of Machine Translation (AAMT) Journal, N0.22, Tokyo, Japan, pages 12-20.

PF Brown, SA Della Pietra, VJ Della Pietra, RL Mercer.1992.The Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics. 19(2):263-309.

Marine Carpuat. 2009. One Translation per Discourse. In Proc. of the NAACL HLT workshop on Semantic Evaluation, pages 19-26.

John DeNero, Alexandre Buchard-Cˆot´e, and Dan Klein. 2008. Sampling Alignment Structure under a Bayesian Translation Model. In Proc. of EMNLP 2008, pages 314–323, Honolulu, October.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One Sense per Discourse. In Proceedings of the workshop on Speech and Natural Language, Harriman, NY.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel,and Alex Waibel. 2005. Adaptation of the Translation Model for Statistical Machine Translation based on Information Retrieval. Proceedings of EAMT 2005:133-142.

Rukmini M. Iyer and Mari Ostendorf. 1999. Modeling Long Distance Dependence in Language:Topic Mixtures Versus Dynamic Cache Models. IEEE Transactions on speech and audio processing, 7(1).

Philipp Koehn, Franz Josef Och ,and DanielMarcu.2003. Statistical Phrase-Based Translation. Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, pages 48-54.

Philipp Koehn.2004.Statistical Significance Tests for Machine Translation Evaluation. In Proc. of EMNLP 2004, pages 388–395.

Roland Kuhn and Renato De Mori. 1990. A Cache-based Natural Language Model for Speech Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence,12(6):570-583.

Yajuan Lu, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In Proc. of EMNLP 2007,pages 343–350, Prague, Czech Republic, June.

Daniel Marcu and William Wong. 2002. A phrase-based Joint Probability Model for Statistical Machine Translation. In Proc. of EMNLP 2002, July.

Laurent Nepveu, Guy Lapalme, Philippe Langlais and George Foster.2004. Adaptive Language and Translation Models for Interactive Machine Translation. In Proc. of EMNLP 2004, pages 190-197.

Eugene A. Nida. 1964. Toward a Science of Translating. Leiden, Netherlands:E.J.Brill.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In Proc. of ACL, pages 160–167.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In Proc. of ACL, pages 440–447.

Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In Proc. of ACL02, pages 311–318.

Martin Raab. 2007. Language Modeling for Machine Translation. VDM Verlag, Saarbrucken, Germany.

G. Salton and C. Buckley. 1988. Term-weighting Approaches in Automatic Text Retrieval. Information Processing and management,24(5):513-523,1988.

Yik-Cheung Tam, Ian Lane and Tanja Schultz. 2007. Bilingual ISA-based Adaptation for Statistical Machine Translation. *Machine Translation*, 28:187-207.

Jorg Tiedemann. 2010. Context Adaptation in Statistical Machine Translation Using Models with Exponentially Decaying Cache. In Proc. of the 2010 workshop on domain adaptation for Natural Language Processing, ACL 2010, pages 8-15.

Joern Wuebker and Arne Mauser and Hermann Ney. 2010. Training Phrase Translation Models with Leaving-One-Out. In Proc. of ACL, pages 475-484.

Bing Zhao, Matthias Eck, and Stephan Vogel. 2004.Language Model Adaptation for Statistical Machine Translation with Structured Query Models. In COLING 2004, Geneva, August.

Bing Zhao and Eric P. Xing .2006. BiTAM:Bilingual Topic Ad-Mixture Models for Word Alignment. In Proc. of ACL2006.

Minimum Imputed Risk: Unsupervised Discriminative Training for Machine Translation

Zhifei Li*
Google Research
Mountain View, CA 94043, USA
zhifei.work@gmail.com

Jason Eisner
Johns Hopkins University
Baltimore, MD 21218, USA
eisner@jhu.edu

Ziyuan Wang, Sanjeev Khudanpur
Johns Hopkins University
Baltimore, MD 21218, USA
zwang40,khudanpur@jhu.edu

Brian Roark
Oregon Health & Science University
Beaverton, Oregon 97006, USA
roark@cslu.ogi.edu

Abstract

Discriminative training for machine translation has been well studied in the recent past. A limitation of the work to date is that it relies on the availability of high-quality in-domain bilingual text for supervised training. We present an unsupervised discriminative training framework to incorporate the usually plentiful target-language monolingual data by using a rough "reverse" translation system. Intuitively, our method strives to ensure that probabilistic "round-trip" translation from a target-language sentence to the source-language and back will have low expected loss. Theoretically, this may be justified as (discriminatively) minimizing an *imputed empirical risk*. Empirically, we demonstrate that augmenting supervised training with unsupervised data improves translation performance over the supervised case for both IWSLT and NIST tasks.

1 Introduction

Missing data is a common problem in statistics when fitting the parameters θ of a model. A common strategy is to attempt to impute, or "fill in," the missing data (Little and Rubin, 1987), as typified by the EM algorithm. In this paper we develop imputation techniques when θ is to be trained *discriminatively*.

We focus on machine translation (MT) as our example application. A Chinese-to-English machine translation system is given a Chinese sentence x and asked to predict its English translation y. This system employs statistical models $p_\theta(y \mid x)$ whose parameters θ are discriminatively trained using bilingual sentence pairs (x, y). But bilingual data for such supervised training may be relatively scarce for a particular language pair (e.g., Urdu-English), especially for some topics (e.g., technical manuals) or genres (e.g., blogs). So systems seek to exploit additional monolingual data, i.e., a corpus of English sentences y with no corresponding source-language sentences x, to improve estimation of θ. This is our missing data scenario.[1]

Discriminative training of the parameters θ of $p_\theta(y \mid x)$ using monolingual English data is a curious idea, since there is no Chinese input x to translate. We propose an unsupervised training approach, called *minimum imputed risk training*, which is conceptually straightforward: *First guess x (probabilistically) from the observed y using a reverse English-to-Chinese translation model $p_\phi(x \mid y)$. Then train the discriminative Chinese-to-English model $p_\theta(y \mid x)$ to do a good job at translating this imputed x back to y, as measured by a given performance metric.* Intuitively, our method strives to ensure that probabilistic "round-trip" translation from a target-language sentence to the source-language and back again will have low expected loss.

Our approach can be applied in an application scenario where we have (1) enough out-of-domain bilingual data to build two baseline translation systems, with parameters θ for the forward direction, and ϕ for the reverse direction; (2) a small amount

* Zhifei Li is currently working at Google Research, and this work was done while he was a PHD student at Johns Hopkins University.

[1] Contrast this with traditional semi-supervised training that looks to exploit "unlabeled" inputs x, with missing *outputs* y.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 920–929,
Edinburgh, Scotland, UK, July 27–31, 2011.

of in-domain bilingual development data to discriminatively tune a *small* number of parameters in ϕ; and (3) a large amount of in-domain English monolingual data.

The novelty here is to exploit (3) to *discriminatively* tune the parameters θ of all *translation* model components,[2] $p_\theta(y|x)$ and $p_\theta(y)$, not merely train a generative *language* model $p_\theta(y)$, as is the norm.

Following the theoretical development below, the empirical effectiveness of our approach is demonstrated by replacing a key *supervised* discriminative training step in the development of large MT systems — learning the log-linear combination of several component model scores (viewed as features) to optimize a performance metric (e.g. BLEU) on a set of (x, y) pairs — with our *unsupervised* discriminative training using only y. One may hence contrast our approach with the traditional *supervised* methods applied to the MT task such as minimum error rate training (Och, 2003; Macherey et al., 2008), the averaged Perceptron (Liang et al., 2006), maximum conditional likelihood (Blunsom et al., 2008), minimum risk (Smith and Eisner, 2006; Li and Eisner, 2009), and MIRA (Watanabe et al., 2007; Chiang et al., 2009).

We perform experiments using the open-source MT toolkit **Joshua** (Li et al., 2009a), and show that adding unsupervised data to the traditional supervised training setup improves performance.

2 Supervised Discriminative Training via Minimization of Empirical Risk

Let us first review discriminative training in the *supervised* setting—as used in MERT (Och, 2003) and subsequent work.

One wishes to tune the parameters θ of some complex translation system $\delta_\theta(x)$. The function δ_θ, which translates Chinese x to English $y = \delta_\theta(x)$ need not be probabilistic. For example, θ may be the parameters of a scoring function used by δ, along with pruning and decoding heuristics, for extracting a high-scoring translation of x.

The goal of discriminative training is to minimize the expected loss of $\delta_\theta(\cdot)$, under a given task-specific **loss function** $\mathrm{L}(y', y)$ that measures how bad it would be to output y' when the correct output is y. For an MT system that is judged by the BLEU metric (Papineni et al., 2001), for instance, $\mathrm{L}(y', y)$ may be the negated BLEU score of y' w.r.t. y. To be precise, the goal[3] is to find θ with low **Bayes risk**,

$$\theta^* = \operatorname*{argmin}_\theta \sum_{x,y} p(x, y)\, \mathrm{L}(\delta_\theta(x), y) \qquad (1)$$

where $p(x, y)$ is the joint distribution of the input-output pairs.[4]

The true $p(x, y)$ is, of course, not known and, in practice, one typically minimizes **empirical risk** by replacing $p(x, y)$ above with the empirical distribution $\tilde{p}(x, y)$ given by a supervised training set $\{(x_i, y_i), i = 1, \ldots, N\}$. Therefore,

$$\begin{aligned} \theta^* &= \operatorname*{argmin}_\theta \sum_{x,y} \tilde{p}(x, y)\, \mathrm{L}(\delta_\theta(x), y) \\ &= \operatorname*{argmin}_\theta \frac{1}{N} \sum_{i=1}^{N} \mathrm{L}(\delta_\theta(x_i), y_i). \qquad (2) \end{aligned}$$

The search for θ^* typically requires the use of numerical methods and some regularization.[5]

3 Unsupervised Discriminative Training with Missing Inputs

3.1 Minimization of Imputed Risk

We now turn to the unsupervised case, where we have training examples $\{y_i\}$ but not their corresponding inputs $\{x_i\}$. We cannot compute the summand $\mathrm{L}(\delta_\theta(x_i), y_i)$ for such i in (2), since $\delta_\theta(x_i)$ requires to know x_i. So we propose to replace

[2] Note that the extra monolingual data is used only for tuning the model weights, but not for inducing new phrases or rules.

[3] This goal is different from the minimum risk training of Li and Eisner (2009) in a subtle but important way. In both cases, θ^* minimizes *risk* or *expected loss*, but the expectation is w.r.t. different distributions: the expectation in Li and Eisner (2009) is under the conditional distribution $p(y \mid x)$, while the expectation in (1) is under the joint distribution $p(x, y)$.

[4] In the terminology of statistical decision theory, $p(x, y)$ is a distribution over states of nature. We seek a *decision rule* $\delta_\theta(x)$ that will incur low expected loss on *observations* x that are generated from unseen states of nature.

[5] To compensate for the shortcut of using the unsmoothed empirical distribution rather than a posterior estimate of $p(x, y)$ (Minka, 2000), it is common to add a regularization term $||\theta||_2^2$ in the objective of (2). The regularization term can prevent overfitting to a training set that is not large enough to learn all parameters.

$\mathrm{L}(\delta_\theta(x_i), y_i)$ with the expectation

$$\sum_x p_\phi(x \mid y_i)\, \mathrm{L}(\delta_\theta(x), y_i), \quad (3)$$

where $p_\phi(\cdot \mid \cdot)$ is a "reverse prediction model" that attempts to impute the missing x_i data. We call the resulting variant of (2) the minimization of *imputed empirical risk*, and say that

$$\theta^* = \operatorname*{argmin}_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid y_i)\, \mathrm{L}(\delta_\theta(x), y_i) \quad (4)$$

is the estimate with the **minimum imputed risk**[6].

The minimum imputed risk objective of (4) could be evaluated by *brute force* as follows.

1. For each unsupervised example y_i, use the reverse prediction model $p_\phi(\cdot \mid y_i)$ to impute possible reverse translations $\mathcal{X}_i = \{x_{i1}, x_{i2}, \ldots\}$, and add each (x_{ij}, y_i) pair, weighted by $p_\phi(x_{ij} \mid y_i) \leq 1$, to an imputed training set .
2. Perform the supervised training of (2) on the *imputed* and *weighted* training data.

The second step means that we must use δ_θ to forward-translate each imputed x_{ij}, evaluate the loss of the translations y'_{ij} against the corresponding true translation y_i, and choose the θ that minimizes the weighted sum of these losses (i.e., the empirical risk when the empirical distribution $\tilde{p}(x, y)$ is derived from the imputed training set). Specific to our MT task, this tries to ensure that probabilistic "round-trip" translation, from the target-language sentence y_i to the source-language and back again, will have a low expected loss.[7]

The trouble with this method is that the reverse model p_ϕ generates a weighted lattice or hypergraph $\mathcal{X}_i$ encoding exponentially many translations of y_i, and it is computationally infeasible to forward-translate *each* $x_{ij} \in \mathcal{X}_i$. We therefore investigate several approximations to (4) in Section 3.4.

[6] One may exploit both supervised data $\{(x_i, y_i)\}$ and unsupervised data $\{y_j\}$ to perform semi-supervised training via an interpolation of (2) and (4). We will do so in our experiments.

[7] Our approach may be applied to other tasks as well. For example, in a speech recognition task, δ_θ is a speech recognizer that produces text, whereas p_ϕ is a speech *synthesizer* that must produce a distribution over audio (or at least over acoustic features or phone sequences) (Huang et al., 2010).

3.2 The Reverse Prediction Model p_ϕ

A *crucial* ingredient in (4) is the reverse prediction model $p_\phi(\cdot|\cdot)$ that attempts to impute the missing x_i. We will train this model in advance, doing the best job we can from available data, including any out-of-domain bilingual data as well as any in-domain monolingual data[8] x.

In the MT setting, δ_θ and p_ϕ may have similar parameterization. One translates Chinese to English; the other translates English to Chinese.

Yet the setup is not quite symmetric. Whereas δ_θ is a translation *system* that aims to produce a *single*, *low-loss* translation, the reverse version p_ϕ is rather a probabilistic *model*. It is supposed to give an accurate probability distribution over possible values x_{ij} of the missing input sentence x_i. All of these values are taken into account in (4), regardless of the loss that they would incur if they were evaluated for translation quality relative to the missing x_i.

Thus, ϕ does not need to be trained to minimize the risk itself (so there is no circularity). Ideally, it should be trained to match the underlying conditional distribution of x given y, by achieving a low conditional cross-entropy

$$\mathrm{H}(X \mid Y) = -\sum_{x,y} p(x, y) \log p_\phi(x \mid y). \quad (5)$$

In practice, ϕ is trained by (empirically) minimizing $-\frac{1}{M}\sum_{j=1}^{N} \log p_\phi(x_j \mid y_j) + \frac{1}{2\sigma^2}\|\phi\|_2^2$ on some bilingual data, with the regularization coefficient σ^2 tuned on held out data.

It may be tolerable for p_ϕ to impute mediocre translations x_{ij}. All that is necessary is that the (forward) translations generated from the imputed x_{ij} "simulate" the competing hypotheses that we would see when translating the correct Chinese input x_i.

3.3 The Forward Translation System δ_θ and The Loss Function $\mathbf{L}(\delta_\theta(x_i), y_i)$

The minimum empirical risk objective of (2) is quite general and various popular supervised training methods (Lafferty et al., 2001; Collins, 2002; Och, 2003; Crammer et al., 2006; Smith and Eisner,

[8] In a translation task from x to y, one usually does not make use of in-domain monolingual data x. But we *can* exploit x to train a language model $p_\phi(x)$ for the reverse translation system, which will make the imputed x_{ij} look like true Chinese inputs.

2006) can be formalized in this framework by choosing different functions for δ_θ and $\mathrm{L}(\delta_\theta(x_i), y_i)$. The generality of (2) extends to our minimum imputed risk objective of (4). Below, we specify the δ_θ and $\mathrm{L}(\delta_\theta(x_i), y_i)$ we considered in our investigation.

3.3.1 Deterministic Decoding

A simple translation rule would define

$$\delta_\theta(x) = \operatorname*{argmax}_y p_\theta(y \mid x) \tag{6}$$

If this $\delta_\theta(x)$ is used together with a loss function $\mathrm{L}(\delta_\theta(x_i), y_i)$ that is the negated BLEU score[9], our minimum imputed risk objective of (4) is equivalent to MERT (Och, 2003) *on the imputed training data*.

However, this would not yield a differentiable objective function. Infinitesimal changes to θ could result in discrete changes to the winning output string $\delta_\theta(x)$ in (6), and hence to the loss $\mathrm{L}(\delta_\theta(x), y_i)$. Och (2003) developed a specialized line search to perform the optimization, which is not scalable when the number of model parameters θ is large.

3.3.2 Randomized Decoding

Instead of using the argmax of (6), we assume *during training* that $\delta_\theta(x)$ is itself random, i.e. the MT system *randomly* outputs a translation y with probability $p_\theta(y \mid x)$. As a result, we will modify our objective function of (4) to take yet another expectation over the unknown y. Specifically, we will replace $\mathrm{L}(\delta_\theta(x), y_i)$ in (4) with

$$\sum_y p_\theta(y \mid x)\, \mathrm{L}(y, y_i). \tag{7}$$

Now, the minimum imputed empirical risk objective of (4) becomes

$$\theta^* = \operatorname*{argmin}_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_{x,y} p_\phi(x \mid y_i)\, p_\theta(y \mid x)\, \mathrm{L}(y, y_i) \tag{8}$$

If the loss function $\mathrm{L}(y, y_i)$ is a negated BLEU, this is equivalent to performing minimum-risk training described by (Smith and Eisner, 2006; Li and Eisner, 2009) *on the imputed data*.[10]

[9] One can manipulate the loss function to support other methods that use deterministic decoding, such as Perceptron (Collins, 2002) and MIRA (Crammer et al., 2006).

[10] Again, one may manipulate the loss function to support other probabilistic methods that use randomized decoding, such as CRFs (Lafferty et al., 2001).

The objective function in (8) is now differentiable, since each coefficient $p_\theta(y \mid x)$ is a differentiable function of θ, and thus amenable to optimization by gradient-based methods; we use the L-BFGS algorithm (Liu et al., 1989) in our experiments. We perform experiments with the syntax-based MT system **Joshua** (Li et al., 2009a), which implements dynamic programming algorithms for second-order expectation semirings (Li and Eisner, 2009) to efficiently compute the gradients needed for optimizing (8).

3.4 Approximating $p_\phi(x \mid y_i)$

As mentioned at the end of Section 3.1, it is computationally infeasible to forward-translate *each* of the imputed reverse translations x_{ij}. We propose four approximations that are computationally feasible. Each may be regarded as a different approximation of $p_\phi(x \mid y_i)$ in equations (4) or (8).

k-best. For each y_i, add to the imputed training set only the k most probable translations $\{x_{i1}, \ldots x_{ik}\}$ according to $p_\phi(x \mid y_i)$. (These can be extracted from $\mathcal{X}_i$ using standard algorithms (Huang and Chiang, 2005).) Rescale their probabilities to sum to 1.

Sampling. For each y_i, add to the training set k independent samples $\{x_{i1}, \ldots x_{ik}\}$ from the distribution $p_\phi(x \mid y_i)$, each with weight $1/k$. (These can be sampled from $\mathcal{X}_i$ using standard algorithms (Johnson et al., 2007).) This method is known in the literature as *multiple imputation* (Rubin, 1987).

Lattice. [11] Under certain special cases it is be possible to compute the expected loss in (3) exactly via dynamic programming. Although $\mathcal{X}_i$ does contain exponentially many translations, it may use a "packed" representation in which these translations share structure. This representation may furthermore enable sharing work in forward-translation, so as to efficiently translate the entire set $\mathcal{X}_i$ and obtain a distribution over translations y. Finally, the expected loss under that distribution, as required by equation (3), may also be efficiently computable.

All this turns out to be possible if (a) the posterior distribution $p_\phi(x \mid y_i)$ is represented by an *un-*

[11] The lattice approximation is presented here as a theoretical contribution, and we do not empirically evaluate it since its implementation requires extensive engineering effort that is beyond the main scope of this paper.

ambiguous weighted finite-state automaton $\mathcal{X}_i$, (b) the forward translation system δ_θ is structured in a certain way as a weighted synchronous context-free grammar, and (c) the loss function decomposes in a certain way. We omit the details of the construction as beyond the scope of this paper.

In our experimental setting described below, (b) is true (using **Joshua**), and (c) is true (since we use a loss function presented by Tromble et al. (2008) that is an approximation to BLEU and is decomposable). While (a) is not true in our setting because $\mathcal{X}_i$ is a hypergraph (which is ambiguous), Li et al. (2009b) show how to *approximate* a hypergraph representation of $p_\phi(x \mid y_i)$ by an unambiguous WFSA. One could then apply the construction to this WFSA[12], obtaining an approximation to (3).

Rule-level Composition. Intuitively, the reason why the structure-sharing in the hypergraph $\mathcal{X}_i$ (generated by the reverse system) cannot be exploited during forward translating is that when the forward Hiero system translates a string $x_i \in \mathcal{X}_i$, it must parse it into recursive phrases.

But the structure-sharing within the hypergraph of $\mathcal{X}_i$ has already parsed x_i into recursive phrases, in a way determined by the reverse Hiero system; each translation phrase (or rule) corresponding to a hyperedge. To exploit structure-sharing, we can use a forward translation system that decomposes according to that existing parse of x_i. We can do that by considering *only* forward translations that respect the hypergraph structure of $\mathcal{X}_i$. The simplest way to do this is to require complete isomorphism of the SCFG trees used for the reverse and forward translations. In other words, this does round-trip imputation (i.e., from y to x, and then to y') at the rule level. This is essentially the approach taken by Li et al. (2010).

3.5 The Log-Linear Model p_θ

We have not yet specified the form of p_θ. Following much work in MT, we begin with a linear model

$$\text{score}(x, y) = \theta \cdot f(x, y) = \sum_k \theta_k f_k(x, y) \quad (9)$$

where $f(x, y)$ is a feature vector indexed by k. Our deterministic *test-time* translation system δ_θ simply outputs the highest-scoring y for fixed x. At *training time*, our randomized decoder (Section 3.3.2) uses the Boltzmann distribution (here a log-linear model)

$$p_\theta(y \mid x) = \frac{e^{\gamma \cdot \text{score}(x,y)}}{Z(x)} = \frac{e^{\gamma \cdot \text{score}(x,y)}}{\sum_{y'} e^{\gamma \cdot \text{score}(x,y')}} \quad (10)$$

The scaling factor γ controls the sharpness of the training-time distribution, i.e., the degree to which the randomized decoder favors the highest-scoring y. For large γ, our training objective approaches the imputed risk of the deterministic test-time system while remaining differentiable.

In a task like MT, in addition to the input x and output y, we often need to introduce a *latent* variable d to represent the hidden derivation that relates x to y. A derivation d represents a particular *phrase segmentation* in a phrase-based MT system (Koehn et al., 2003) and a *derivation tree* in a typical syntax-based system (Galley et al., 2006; Chiang, 2007). We change our model to assign scores not to an (x, y) pair but to the detailed derivation d; in particular, now the function f that extracts a feature vector can look at all of d. We replace y by d in (9)–(10), and finally define $p_\theta(y|x)$ by marginalizing out d,

$$p_\theta(y \mid x) = \sum_{d \in \mathrm{D}(x,y)} p_\theta(d \mid x) \quad (11)$$

where $\mathrm{D}(x, y)$ represents the set of derivations that yield x and y.

4 Minimum Imputed Risk vs. EM

The notion of imputing missing data is familiar from other settings (Little and Rubin, 1987), particularly the expectation maximization (EM) algorithm, a widely used generative approach. So it is instructive to compare EM with minimum imputed risk.

One can estimate θ by maximizing the log-likelihood of the data $\{(x_i, y_i), i = 1, \ldots, N\}$ as

$$\operatorname*{argmax}_\theta \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x_i, y_i). \quad (12)$$

If the x_i's are missing, EM tries to iteratively maximize the marginal probability:

$$\operatorname*{argmax}_\theta \frac{1}{N} \sum_{i=1}^{N} \log \sum_x p_\theta(x, y_i). \quad (13)$$

[12] Note that the forward translation of a WFSA is tractable by using a lattice-based decoder such as that by Dyer et al. (2008).

The E-step of each iteration comprises computing $\sum_x p_{\theta_t}(x \mid y_i) \log p_\theta(x, y_i)$, the *expected* log-likelihood of the complete data, where $p_{\theta_t}(x \mid y_i)$ is the conditional part of $p_{\theta_t}(x, y_i)$ under the current iterate θ_t, and the M-step comprises maximizing it:

$$\theta_{t+1} = \operatorname*{argmax}_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_{\theta_t}(x \mid y_i) \log p_\theta(x, y_i). \quad (14)$$

Notice that if we replace $p_{\theta_t}(x|y_i)$ with $p_\phi(x \mid y_i)$ in the equation above, and admit negated log-likelihood as a loss function, then the EM update (14) becomes identical to (4). In other words, the minimum imputed risk approach of Section 3.1 differs from EM in (i) using an externally-provided and static p_ϕ, instead of refining it at each iteration based on the current p_{θ_t}, and (ii) using a specific loss function, namely negated log-likelihood.

So why not simply use the maximum-likelihood (EM) training procedure for MT? One reason is that it is not discriminative: the loss function (e.g. negated BLEU) is ignored during training.

A second reason is that training good *joint* models $p_\theta(x, y)$ is computationally expensive. Contemporary MT makes heavy use of log-linear probability models, which allow the system designer to inject phrase tables, linguistic intuitions, or prior knowledge through a careful choice of features. Computing the objective function of (14) in closed form is difficult if p_θ is an arbitrary log-linear model, because the joint probability $p_\theta(x_i, y_i)$ is then defined as a ratio whose denominator Z_θ involves a sum over all possible sentence pairs (x, y) of any length.

By contrast, our discriminative framework will only require us to work with conditional models. While conditional probabilities such as $p_\phi(x \mid y)$ and $p_\theta(y \mid x)$ are also ratios, computing their denominators only requires us to sum over a packed forest of possible translations of a given y or x.[13]

In summary, EM would impute missing data using $p_\theta(x \mid y)$ and predict outputs using $p_\theta(y \mid x)$, both being conditional forms of the same joint model $p_\theta(x, y)$. Our minimum imputed risk training method is similar, but it instead uses a pair of separately parameterized, separately trained models $p_\phi(x \mid y)$ and $p_\theta(y \mid x)$. By sticking to conditional models, we can efficiently use more sophisticated model features, and we can incorporate the loss function when we train θ, which should improve both efficiency and accuracy at test time.

[13] Analogously, discriminative CRFs have become more popular than generative HMMs because they permit efficient training even with a wide variety of log-linear features (Lafferty et al., 2001).

5 Experimental Results

We report results on Chinese-to-English translation tasks using **Joshua** (Li et al., 2009a), an open-source implementation of Hiero (Chiang, 2007).

5.1 Baseline Systems

5.1.1 IWSLT Task

We train both reverse and forward baseline systems. The translation models are built using the corpus for the IWSLT 2005 Chinese to English translation task (Eck and Hori, 2005), which comprises 40,000 pairs of transcribed utterances in the travel domain. We use a *5-gram* language model with modified Kneser-Ney smoothing (Chen and Goodman, 1998), trained on the English (resp. Chinese) side of the bitext. We use a standard training pipeline and pruning settings recommended by (Chiang, 2007).

5.1.2 NIST Task

For the NIST task, the TM is trained on about 1M parallel sentence pairs (about 28M words in each language), which are sub-sampled from corpora distributed by LDC for the NIST MT evaluation using a sampling method implemented in **Joshua**. We also used a 5-gram language model, trained on a data set consisting of a 130M words in English Gigaword (LDC2007T07) and the bitext's English side.

5.2 Feature Functions

We use two classes of features f_k for discriminative training of p_θ as defined in (9).

5.2.1 Regular Hiero Features

We include ten features that are standard in Hiero (Chiang, 2007). In particular, these include one baseline language model feature, three baseline translation models, one word penalty feature, three features to count how many rules with an arity of

zero/one/two are used in a derivation, and two features to count how many times the unary and binary glue rules in Hiero are used in a derivation.

5.2.2 Target-rule Bigram Features

In this paper, we do not attempt to discriminatively tune a separate parameter for each bilingual rule in the Hiero grammar. Instead, we train several hundred features that generalize across these rules.

For each bilingual rule, we extract bigram features over the target-side symbols (including non-terminals and terminals). For example, if a bilingual rule's target-side is "*on the* X_1 *issue of* X_2" where X_1 and X_2 are non-terminals (with a position index), we extract the bigram features *on the*, *the* X, X *issue*, *issue of*, and *of* X. (Note that the position index of a non-terminal is ignored in the feature.) Moreover, for the terminal symbols, we will use their dominant POS tags (instead of the symbol itself). For example, the feature *the* X becomes *DT* X. We use 541 such bigram features for IWSLT task (and 1023 such features for NIST task) that fire frequently.

5.3 Data Sets for Discriminative Training

5.3.1 IWSLT Task

In addition to the 40,000 sentence pairs used to train the baseline generative models (which are used to compute the features f_k), we use three bilingual data sets listed in Table 1, also from IWSLT, for discriminative training: one to train the reverse model p_ϕ (which uses only the 10 standard Hiero features as described in Section 5.2.1),[14] one to train the forward model δ_θ (which uses both classes of features described in Section 5.2, i.e., 551 features in total), and one for test.

Note that the reverse model ϕ is always trained using the supervised data of Dev_ϕ, while the forward model θ may be trained in a supervised or semi-supervised manner, as we will show below.

In all three data sets, each Chinese sentence x_i has 16 English reference translations, so each y_i is actually a *set* of 16 translations. When we impute data from y_i (in the semi-supervised scenario), we actually impute 16 different values of x_i, by using p_ϕ to separately reverse translate each sentence in y_i. This effectively adds 16 pairs of the form (x_i, y_i) to the training set (see section 3.4), where each x_i is a different input sentence (imputed) in each case, but y_i is always the original set of 16 references.

Data set	Purpose	# of sentences	
		Chinese	English
Dev_ϕ	training ϕ	503	503×16
Dev_θ	training θ	503*	503×16
Eval_θ	testing	506	506×16

Table 1: **IWSLT Data sets used for discriminative training/test**. Dev_ϕ is used for discriminatively training of the reverse model ϕ, Dev_θ is for the forward model, and Eval_θ is for testing. The star * for Dev_θ emphasizes that some of its Chinese side will not be used in the training (see Table 2 for details).

5.3.2 NIST Task

For the NIST task, we use MT03 set (having 919 sentences) to tune the component parameters in both the forward and reverse baseline systems. Additionally, we use the English side of MT04 (having 1788 sentences) to perform semi-supervised tuning of the forward model. The test sets are MT05 and MT06 (having 1082 and 1099 sentences, respectively). In all the data sets, each source sentence has four reference translations.

5.4 Main Results

We compare two training scenarios: supervised and semi-supervised. The supervised system ("Sup") carries out discriminative training on a bilingual data set. The semi-supervised system ("+Unsup") additionally uses some monolingual English text for discriminative training (where we impute one Chinese translation per English sentence).

Tables 2 and 3 report the results for the two tasks under two training scenarios. Clearly, adding unsupervised data improves over the supervised case, by at least 1.3 BLEU points in IWSLT and 0.5 BLEU in NIST.

5.5 Results for Analysis Purposes

Below, we will present more results on the IWSLT data set to help us understand the behavior of the

[14] Ideally, we should train ϕ to minimize the conditional cross-entropy (5) as suggested in section 3.2. In the present results, we trained ϕ discriminatively to minimize risk, purely for ease of implementation using well versed steps.

Training scenario	Test BLEU
Sup, (200, 200×16)	47.6
+Unsup, 101×16 Eng sentences	49.0
+Unsup, 202×16 Eng sentences	48.9
+Unsup, 303×16 Eng sentences	49.7*

Table 2: **BLEU scores for semi-supervised training for IWSLT task.** The supervised system ("Sup") is trained on a subset of Dev_θ containing 200 Chinese sentences and 200×16 English translations. "+Unsup" means that we include additional (monolingual) English sentences from Dev_θ for semi-supervised training; for each English sentence, we impute the 1-best Chinese translation. A star * indicates a result that is signicantly better than the "Sup" baseline (paired permutation test, $p < 0.05$).

Training scenario	Test BLEU	
	MT05	MT06
Sup, (919, 919×4)	32.4	30.6
+Unsup, 1788 Eng sentences	33.0*	31.1*

Table 3: **BLEU scores for semi-supervised training for NIST task.** The "Sup" system is trained on MT03, while the "+Unsup" system is trained with additional 1788 English sentences from MT04. (Note that while MT04 has 1788×4 English sentences as it has four sets of references, we only use one such set, for computational efficiency of discriminative training.) A star * indicates a result that is signicantly better than the "Sup" baseline (paired permutation test, $p < 0.05$).

methods proposed in this paper.

5.5.1 Imputation with Different Reverse Models

A critical component of our unsupervised method is the reverse translation model $p_\phi(x \mid y)$. We wonder how the performance of our unsupervised method changes when the quality of the reverse system varies. To study this question, we used two different reverse translation systems, one with a language model trained on the Chinese side of the bitext ("WLM"), and the other one without using such a Chinese LM ("NLM"). Table 4 (in the fully unsupervised case) shows that the imputed Chinese translations have a far lower BLEU score without the language model,[15] and that this costs us about 1 English BLEU point in the forward translations. Still, even with the worse imputation (in the case of "NLM"), our forward translations improve as we add more monolingual data.

Data size	Imputed-CN BLEU		Test-EN BLEU	
	WLM	NLM	WLM	NLM
101	11.8	3.0	48.5	46.7
202	11.7	3.2	48.9	47.6
303	13.4	3.5	48.8	47.9

Table 4: **BLEU scores for unsupervised training with/without using a language model in the reverse system.** A data size of 101 means that we use only the English sentences from a subset of Dev_θ containing 101 Chinese sentences and 101×16 English translations; for each English sentence we impute the 1-best Chinese translation. "WLM" means a Chinese language model is used in the reverse system, while "NLM" means no Chinese language model is used. In addition to reporting the BLEU score on Eval_θ, we also report "Imputed-CN BLEU", the BLEU score of the imputed Chinese sentences against their corresponding Chinese reference sentences.

5.5.2 Imputation with Different k-best Sizes

In all the experiments so far, we used the reverse translation system to impute only a single Chinese translation for each English monolingual sentence. This is the 1-best approximation of section 3.4.

Table 5 shows (in the fully unsupervised case) that the performance does not change much as k increases.[16] This may be because that the 5-best sentences are likely to be quite similar to one another (May and Knight, 2006). Imputing a longer k-best list, a sample, or a lattice for x_i (see section 3.4) might achieve more diversity in the training inputs, which might make the system more robust.

6 Conclusions

In this paper, we present an unsupervised discriminative training method that works with missing inputs. The key idea in our method is to use a reverse model to impute the missing input from the observed output. The training will then forward translate the imputed input, and choose the parameters of the forward model such that the imputed risk (i.e.,

[15] The BLEU scores are low even *with* the language model because only one Chinese reference is available for scoring.

[16] In the present experiments, however, we simply weighted all k imputed translations equally, rather than in proportion to their posterior probabilities as suggested in Section 3.4.

Training scenario	Test BLEU
Unsup, k=1	48.5
Unsup, k=2	48.4
Unsup, k=3	48.9
Unsup, k=4	48.5
Unsup, k=5	48.4

Table 5: **BLEU scores for unsupervised training with different k-best sizes.** We use 101×16 monolingual English sentences, and for each English sentence we impute the k-best Chinese translations using the reverse system.

the expected loss of the forward translations with respect to the observed output) is minimized. This matches the intuition that the probabilistic "round-trip" translation from the target-language sentence to the source-language and back should have low expected loss.

We applied our method to two Chinese to English machine translation tasks (i.e. IWSLT and NIST). We showed that augmenting supervised data with unsupervised data improved performance over the supervised case (for both tasks).

Our discriminative model used only a small amount of training data and relatively few features. In future work, we plan to test our method in settings where there are large amounts of monolingual training data (enabling many discriminative features). Also, our experiments here were performed on a language pair (i.e., Chinese to English) that has quite rich bilingual resources in the domain of the test data. In future work, we plan to consider low-resource test domains and language pairs like Urdu-English, where bilingual data for novel domains is sparse.

Acknowledgements

This work was partially supported by NSF Grants No IIS-0963898 and No IIS-0964102 and the DARPA GALE Program. The authors thank Markus Dreyer, Damianos Karakos and Jason Smith for insightful discussions.

References

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *ACL*, pages 200–208.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *NAACL*, pages 218–226.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *ACL*, pages 1012–1020.

Matthias Eck and Chiori Hori. 2005. Overview of the iwslt 2005 evaluation campaign. In *In IWSLT*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *ACL*, pages 961–968.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *IWPT*, pages 53–64.

Jui-Ting Huang, Xiao Li, and Alex Acero. 2010. Discriminative training methods for language models using conditional entropy criteria. In *ICASSP*.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *NAACL*, pages 139–146.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*, pages 48–54.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *EMNLP*, pages 40–51.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar. Zaidan. 2009a. Joshua: An open source toolkit for parsing-based machine translation. In *WMT09*, pages 26–30.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009b. Variational decoding for statistical machine translation. In *ACL*, pages 593–601.

Zhifei Li, Ziyuan Wang, Sanjeev Khudanpur, and Jason Eisner. 2010. Unsupervised discriminative language

model training for machine translation using simulated confusion sets. In *COLING*, pages 556–664.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL*, pages 761–768.

R. J. A. Little and D. B. Rubin. 1987. *Statistical Analysis with Missing Data*. J. Wiley & Sons, New York.

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

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *EMNLP*, pages 725–734.

Jonathan May and Kevin Knight. 2006. A better n-best list: practical determinization of weighted finite tree automata. In *NAACL*, pages 351–358.

Thomas Minka. 2000. Empirical risk minimization is an incomplete inductive principle. In *MIT Media Lab note*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

D. B. Rubin. 1987. *Multiple Imputation for Nonresponse in Surveys*. J. Wiley & Sons, New York.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *ACL*, pages 787–794.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum-Bayes-risk decoding for statistical machine translation. In *EMNLP*, pages 620–629.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *EMNLP-CoNLL*, pages 764–773.

Improving Bilingual Projections via Sparse Covariance Matrices

Jagadeesh Jagarlamudi
University of Maryland
College Park, USA
jags@umiacs.umd.edu

Raghavendra Udupa
Microsoft Research
Bangalore, India
raghavu@microsoft.com

Hal Daumé III
University of Maryland
College Park, USA
hal@umiacs.umd.edu

Abhijit Bhole
Microsoft Research
Bangalore, India
v-abbhol@microsoft.com

Abstract

Mapping documents into an interlingual representation can help bridge the language barrier of cross-lingual corpora. Many existing approaches are based on word co-occurrences extracted from aligned training data, represented as a covariance matrix. In theory, such a covariance matrix should represent semantic equivalence, and *should* be highly sparse. Unfortunately, the presence of noise leads to dense covariance matrices which in turn leads to suboptimal document representations. In this paper, we explore techniques to recover the desired sparsity in covariance matrices in two ways. First, we explore word association measures and bilingual dictionaries to weigh the word pairs. Later, we explore different selection strategies to remove the noisy pairs based on the association scores. Our experimental results on the task of aligning comparable documents shows the efficacy of sparse covariance matrices on two data sets from two different language pairs.

1 Introduction

Aligning documents from different languages arises in a range of tasks such as parallel phrase extraction (Gale and Church, 1991; Rapp, 1999), mining translations for out-of-vocabulary words for statistical machine translation (Daume III and Jagarlamudi, 2011) and document retrieval (Ballesteros and Croft, 1996; Munteanu and Marcu, 2005). In this task, we are given a comparable corpora and some documents in one language are assumed to have a comparable document in the other language and the goal is to recover this hidden alignment. In this paper, we address this problem by mapping the documents into a common subspace (interlingual representation). This common subspace generalizes the notion of vector space model for cross-lingual applications (Turney and Pantel, 2010).

Most of the existing approaches use manually aligned document pairs to find a common subspace in which the aligned document pairs are maximally correlated. The sub-space can be found using either generative approaches based on topic modeling (Mimno et al., 2009; Jagarlamudi and Daumé III, 2010; Zhang et al., 2010; Vu et al., 2009) or discriminative approaches based on variants of Principal Component Analysis (PCA) and Canonical Correlation Analysis (CCA) (Susan T. Dumais, 1996; Vinokourov et al., 2003; Platt et al., 2010; Haghighi et al., 2008). Both styles rely on document level term co-occurrences to find the latent representation.

The discriminative approaches capture essential word co-occurrences in terms of two monolingual covariance matrices and a cross-covariance matrix. Subsequently, they use these covariance matrices to find projection directions in each language such that aligned documents lie close to each other (Sec. 2). The strong reliance of these approaches on the covariance matrices leads to problems, especially with the noisy data caused either by the noisy words in a document or the noisy document alignments. Noisy data is not uncommon and is usually the case with data collected from community based resources such as Wikipedia. This degrades performance of a

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 930–940,
Edinburgh, Scotland, UK, July 27–31, 2011.

variety of tasks, such as transliteration Mining (Klementiev and Roth, 2006; Hermjakob et al., 2008; Ravi and Knight, 2009) and multilingual web search (Gao et al., 2009).

In this paper, we address the problem of identifying and removing noisy entries in the covariance matrices. We address this problem in two stages. In the first stage, we explore the use of word association measures such as Mutual Information (MI) and Yule's ω (Reis and Judd, 2000) in computing the strength of a word pair (Sec. 3.1). We also explore the use of bilingual dictionaries developed from cleaner resources such as parallel data. In the second stage, we use the association strengths in filtering out the noisy word pairs from the covariance matrices. We pose this as a word pair selection problem and explore multiple strategies (Sec. 3.2).

We evaluate the utility of sparse covariance matrices in improving the bilingual projections incrementally (Sec. 4). We first report results on synthetic multi-view data where the true correspondences between features of different views are available. Moreover, this also lets us systematically explore the effect of noise level on the accuracy. Our experimental results show a significant improvement when the true correspondences are available. Later, we report our experimental results on the document alignment task on Europarl and Wikipedia data sets and on two language pairs. We found that sparsifying the covariance matrices helps in general, but using cleaner resource such bilingual dictionaries performed best.

2 Canonical Correlation Analysis (CCA)

In this section, we describe how Canonical Correlation Analysis is used to solve the problem of aligning bilingual documents. We mainly focus on representing the solution of CCA in terms of covariance matrices. Since most of the existing discriminative approaches are variants of CCA, showing the advantage of recovering sparseness in CCA makes it applicable to the other variants as well.

Given a training data of n aligned document pairs, CCA finds projection directions for each language, so that the documents when projected along these directions are maximally correlated (Hotelling, 1936). Let X ($d_1 \times n$) and Y ($d_2 \times n$) be the representation of data in both the languages and further assume that the data is centered (subtract the mean vector from each document *i.e.* $\mathbf{x}_i \leftarrow \mathbf{x}_i - \mu_x$ and $\mathbf{y}_i \leftarrow \mathbf{y}_i - \mu_y$). Then CCA finds projection directions $\mathbf{a}$ and $\mathbf{b}$ which maximize:

$$\frac{\mathbf{a}^T XY^T \mathbf{b}}{\sqrt{\mathbf{a}^T XX^T \mathbf{a}}\sqrt{\mathbf{b}^T YY^T \mathbf{b}}}$$
$$\text{s.t. } \mathbf{a}^T XX^T \mathbf{a} = 1 \ \& \ \mathbf{b}^T YY^T \mathbf{b} = 1$$

The projection directions are obtained by solving the generalized eigen system:

$$\begin{bmatrix} 0 & C^{xy} \\ C^{yx} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} (1\text{-}\lambda)C^{xx}+\lambda I & 0 \\ 0 & (1\text{-}\lambda)C^{yy}+\lambda I \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \quad (1)$$

where $C^{xx} = XX^T$, $C^{yy} = YY^T$ are the monolingual covariance matrices, $C^{xy} = XY^T$ is the cross-covariance matrix and λ is the regularization parameter. Using these eigenvectors as columns, we form the projection matrices A and B. These projection matrices are used to map documents in both the languages into interlingual representation.

Given any new pair of documents, their similarity is computed by first mapping them into the lower dimensions space and computing the cosine similarity between their projections. In general, using all the eigenvectors is sub optimal and thus retaining top eigenvectors leads to better generalizability.

3 Covariance Selection

As shown above, the underlying objective function in most of the discriminative approaches is of the form $\mathbf{a}^T XY^T \mathbf{b}$. This can be rewritten as :

$$\begin{aligned} \mathbf{a}^T XY^T \mathbf{b} &= \sum_{k=1}^{n} \langle \mathbf{x}_k, \mathbf{a} \rangle \langle \mathbf{y}_k, \mathbf{b} \rangle \\ &= \sum_{k=1}^{n} \Big(\sum_{i=1}^{d_1} X_{i,k} a_i \cdot \sum_{j=1}^{d_2} Y_{j,k} b_j \Big) \\ &= \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} a_i b_j \Big(\sum_{k=1}^{n} X_{i,k} Y_{j,k} \Big) \\ &= \sum_{i,j=1}^{d_1,d_2} a_i b_j C_{ij}^{xy} \quad (2) \end{aligned}$$

Similarly, the constraints can also be rewritten as $\sum_{i,j=1}^{d_1} a_i a_j C_{ij}^{xx} = 1$ and $\sum_{i,j=1}^{d_2} b_i b_j C_{ij}^{yy} = 1$.

Maximizing this objective function, under the constraints, involves a careful selection of the vectors $\mathbf{a}$ and $\mathbf{b}$ such that $a_i b_j$ is high whenever C_{ij}^{xy} is high. So, every non-zero entry of the cross-covariance matrix restricts the choice of the projection directions. While this may not be a severe problem when the training data is clean, but this is very uncommon especially in the case of high dimensional data like text documents. Moreover, the inherent ambiguity of natural languages increases the chances of seeing a noisy word in any document. Every occurrence of a noisy word will have a non-zero contribution towards the covariance matrix making it dense, which in turn prevents the selection of appropriate projection directions.

In this section, we describe some techniques to recover the sparsity by removing the noisy entries from the covariance matrices. We break this task into two sub problems: computing an association score for every word pair and then using an appropriate strategy to identify the noisy pairs based on their weights. We explore multiple ways to address both the steps in the following two sections. For the sake of convenience and clarity, we describe our techniques in the context of cross-covariance matrix between English and Spanish language pair. But these techniques extend directly to monolingual covariance matrices, and to different language pairs as well.

3.1 Computing Word Pair Association

The first step in filtering out the noisy word co-occurrences is to use an appropriate measure to compute the strength of word pairs (English and Spanish words). This is a well studied problem and several association measures have been proposed in the NLP literature (Dunning, 1993; Inkpen and Hirst, 2002; Moore, 2004). These association measures can be divided into groups based on the statistics they use (Hoang et al., 2009). Here we explore a few of them for sparsifying the cross-covariance matrix.

3.1.1 Covariance

The first option is to use the cross-covariance matrix itself. As noted above, when the data matrix is centered, the cross-covariance of an English word (e_i) with a Spanish word (f_j) is given by $\sum_{k=1}^{n} X_{ik} Y_{jk}$. It measures the strength with which two words co-occur together. This measure uses information about the occurrence of a word pair in aligned documents and doesn't use other statistics such as 'how often this pair *doesn't* co-occur together' and so on.

3.1.2 Mutual Information

Association measures like covariance and Pointwise Mutual Information, which only use the frequency with which a word pair co-occurs, often overestimate the strength of low frequent words (Moore, 2004). On the other hand, measures like Log-likelihood ratio (Dunning, 1993) and Mutual Information (MI) use other statistics like the marginal probabilities of each of the words.

For any two words, e_i and f_j, let n_{11}, n_{10}, n_{01} and n_{00} denote the number of documents in which both the words co-occur, only English word occurs, only Spanish word occurs and none of the words occur. Then the Mutual Information of this word pair is given by:

$$\text{MI}(e_i, f_j) = \frac{1}{n} \sum_{i,j \in \{0,1\}} n_{ij} \log \frac{n_{ij} \times n}{n_i n_j} \quad (3)$$

where n_i and n_j denote the number of documents in which the English and the Spanish word occurs and n is the total number of documents. We treat the occurrence of a word in a document slightly different from others, we treat a word as occurring in a document if it has occurred more than its average frequency in the corpus. Log-likelihood ratio and the MI differ only in terms of the constant they use, so we use only MI in our experiments.

3.1.3 Yule's ω

Yule's ω is another popular association measure used in psychology (Reis and Judd, 2000). It uses same statistics used by Mutual Information but differs in the way in which they are combined. MI converts the frequencies into probabilities before computing the association measure where as Yule's ω uses the observed frequencies directly, and doesn't make any assumptions about the underlying probability distributions. Given the same interpretation of the variables as introduced in the previous section, the Yule's ω is estimated as:

$$\omega = \frac{\sqrt{n_{00} n_{11}} - \sqrt{n_{01} n_{10}}}{\sqrt{n_{00} n_{11}} + \sqrt{n_{01} n_{10}}} \quad (4)$$

This way of combining the frequencies bears similarity with the log-odds ratio.

3.1.4 Bilingual Dictionary

The above three association measures use the same training data that is available to compute the covariance matrices in CCA. Thus, their utility in bringing additional information, which is not captured by the covariance matrices, is arguable (our experiments show that they are indeed helpful). Moreover, they use document level co-occurrence information which is coarse compared to the co-occurrence at sentence level or the translational information provided by a bilingual dictionary. So, we use bilingual dictionaries as our final resource to weigh the word co-occurrences. Notice that, using bilingual information brings in information gleaned from an external corpus.

We use translation tables learnt using Giza++ (Och and Ney, 2003) on Europarl data set. Since the translation tables are asymmetric, we combine translation tables from both the directions. We first use a threshold on the conditional probability to filter out the low probability ones and then convert them into joint probabilities before combining. For each word pair (e_i, f_j), we compute the score as:

$$\frac{1}{2}\Big(P(e_i|f_j)P(f_j) + P(f_j|e_i)P(e_i)\Big)$$

While the first three association measures can also be applied to monolingual data, bilingual dictionary can't be used for weighting monolingual word pairs. So in this case, we use either of the above mentioned techniques for weighting monolingual word pairs.

3.2 Selection Strategies

The next step after computing association measure for all word pairs is to use them in selecting the pairs that need to be retained. In this section, we describe some approaches such as thresholding and matching for the word pair selection.

3.2.1 Thresholding

A straight forward way to remove the noisy word co-occurrences is to zero out the entries of the cross-covariance matrix that are lower than a threshold. To understand the motivation, consider the rewritten objective function of CCA, $\mathbf{a}^T XY^T \mathbf{b} = \sum_{ij} C^{xy}_{ij} a_i b_j$. This is linear in terms of the individual components of the cross-covariance matrix. So, if we want to remove some of the entries of the covariance matrix with minimal change in the value of the objective function, then the optimal choice is to sort the entries of the covariance matrix and filter out the less confident word pairs.

3.2.2 Relative Thresholding

While the thresholding strategy described in the above section is very simple, it is often biased by the frequent words. Since a frequent word co-occurs with other words often, it naturally tends to have high association with most of the other words. As a result, absolute thresholding tends to remove all the less frequent word pairs while leaving the co-occurrences of the frequent words untouched. Eventually, this may lead to zeroing out some of the rows or the columns of the cross-covariance matrix.

To circumvent this, we try thresholding at word level. For every English word, we choose a few Spanish words that have high association and vice versa. Since the nearest neighbour property is asymmetric, we take the union of all the selected word pairs. That is, we retain a word pair, if either the Spanish word is in the top ranked list of the English word or vice versa.

3.2.3 Maximal Matching

Though relative thresholding overcomes the problem of zeroing out entire rows or columns posed by direct thresholding, it is still biased by the frequent words. The high association measure of a frequent English word with many Spanish words, makes it a nearest neighbour for lot of Spanish words. One way to prevent this is to discourage an already selected English word from associating with a new Spanish word. This requires a global knowledge of all the selected pairs and can not be done by looking at the individual words, as is the case with the greedy strategy employed by the relative thresholding.

We use matching to solve this problem. We formulate the selection of the word pairs as a network flow problem (Jagarlamudi et al., 2011). The objective is to select word pairs that have high association measure while constraining each word to be associated with only a few words from other language. Let I_{ij} denote an indicator variable taking a value of

0 or 1 depending on if the word pair (e_i, f_j) is selected or not. We want each word to be associated with k words from other language, *i.e.*$\sum_j I_{ij} = k$ and $\sum_i I_{ij} = k$. Moreover, we want word pairs with high association score to be selected. We can encode this objective and the constraints as the following optimization problem:

$$\arg\max_{I} \sum_{i,j=1}^{d_1,d_2} C_{ij}^{xy} I_{ij} \quad (5)$$
$$\forall i \sum_j I_{ij} = k;\ \forall j \sum_i I_{ij} = k;\ \forall i,j\ I_{ij} \in \{0,1\}$$

If $k = 1$, then this problem reduces to a linear assignment problem and can be solved optimally using the Hungarian algorithm (Jonker and Volgenant, 1987). For other values of k, this can be solved by relaxing the constraint $I_{ij} \in \{0,1\}$ to $0 \leq I_{ij} \leq 1$. The optimal solution of the relaxed problem can be found efficiently using linear programming (Ravindra et al., 1993). The uni-modular nature of the constraints guarantees an integral solution (Schrijver, 2003), so relaxing the original integer problem doesn't introduce any error in the optimal solution.

3.2.4 Monolingual Augmentation

The above three selection strategies operate on the covariance matrices independently. In this section we propose to combine them. Specifically, we propose to augment the set of selected bilingual word pairs using the monolingual word pairs. We first use any of the above mentioned strategies to select bilingual and monolingual word pairs. Let I^{xy}, I^{xx} and I^{yy} be the binary matrices that indicate the selected word pairs based on the bilingual and monolingual association scores. Then the monolingual augmentation strategy updates I^{xy} in the following way:

$$I^{xy} \leftarrow \text{Binarize}(I^{xx} I^{xy} I^{yy})$$

i.e., we multiply I^{xy} with the monolingual selection matrices and then binarize the resulting matrix. Our monolingual augmentation is motivated by the following probabilistic interpretation:

$$P(x,y) = \sum_{x',y'} P(x|x')P(y|y')P(x',y')$$

which can be rewritten as $P \leftarrow T^x P (T^y)^T$ where T^x and T^y are monolingual state transition matrices.

3.3 Our Approach

In this section we summarize our approach for the task of finding aligned documents from a cross-lingual comparable corpora. The training phase involves finding projection directions for documents of both the languages. We compute the covariance matrices using the training data. Then we use any of the word association measures (Sec. 3.1) along with a selection criteria (Sec. 3.2) to recover the sparseness in either only the cross-covariance or all of the covariance matrices. Let I^{xy}, I^{xx} and I^{yy} be the binary matrices which represent the word pairs that are selected based on the chosen sparsification technique. Now, we replace the covariance matrices in Eq. 1 as follows: $C^{xx} \leftarrow C^{xx} \otimes I^{xx}$, $C^{yy} \leftarrow C^{yy} \otimes I^{yy}$ and $C^{xy} \leftarrow C^{xy} \otimes I^{xy}$ where $\otimes$ denotes the element-wise matrix product. Subsequently, we solve the generalized eigenvalue problem shown in Eq. 1 to obtain the projection directions. Let A and B be the matrices formed with top eigenvectors of Eq. 1 as the columns. These projection matrices are used to map documents into the interlingual representation. Such an interlingual representation is useful in many tasks like cross-lingual text categorization (Bel et al., 2003) multilingual web search (Gao et al., 2009) and so on.

During the testing, given an English document $\mathbf{x}$, finding an aligned Spanish document involves solving:

$$\arg\max_{\mathbf{y}} \frac{\mathbf{x}^T \Big((AB^T) \otimes I^{xy} \Big) \mathbf{y}}{\sqrt{\mathbf{x}^T \Big((AA^T) \otimes I^{xx} \Big) \mathbf{x}} \sqrt{\mathbf{y}^T \Big((BB^T) \otimes I^{yy} \Big) \mathbf{y}}}$$

If the documents are normalized before hand, then the above equation reduces to computing only the numerator.

4 Experiments

4.1 Experimental Setup

We experiment with the task of finding aligned documents from a cross-lingual comparable corpora. In this task, we are given comparable corpora consisting of two document collections, each in a different language. As the corpora are comparable, some documents in one collection have a comparable document in the other collection. The task is to recover

this hidden alignment. The recovered alignment is compared against the ground truth.

We evaluate our idea of sparsifying the covariance matrices incrementally. We first evaluate the effectiveness of our approach on synthetic data, as it enables us to systematically study the effect of noise. Subsequently, we evaluate each of the above discussed sparsification strategies on real world data sets. We have discussed four possible ways for computing word association measure and three approaches for word pair selection. That leaves us 12 different ways for sparsifying the covariance matrices, with each method having parameters to control the amount of sparseness. We use a small amount of development data for model selection and parameter tuning and choose a few promising models. Finally, we compare these selected models with state-of-the-art baselines on two language pairs and on two different data sets.

In each case, we use the training data to learn the projection directions. And then, for each of the test documents, we find the aligned document from other language. We report average accuracy of the top ranked document and also the Mean Reciprocal Rank (MRR) of the true aligned document.

4.2 Synthetic Data

We follow the generative story introduced in Bach and Jordan (2005) to generate synthetic multi-view data. Their method does not assume any correspondence between the feature dimensions of both the views. We modify their approach slightly so that we know the actual correspondence between the features. We use these true feature correspondences for sparsification of the cross-covariance matrix.

We first generate a d dimensional vector in the common latent space and then use the projection matrices to map it into the individual feature spaces as follows:

$$z \sim \mathcal{N}(0, I_d)$$
$$x|z \sim (W_1 z + \mu_1) + \eta\, \mathcal{N}(0, I_{d_1})$$
$$y|z \sim (W_1 z + \mu_2) + \eta\, \mathcal{N}(0, I_{d_2})$$

Notice that we use the same projection matrix W_1 for both the views, this ensures a one-to-one correspondence between the features of both the spaces. Moreover, we also introduce a parameter η which controls the amount of noise in the data.

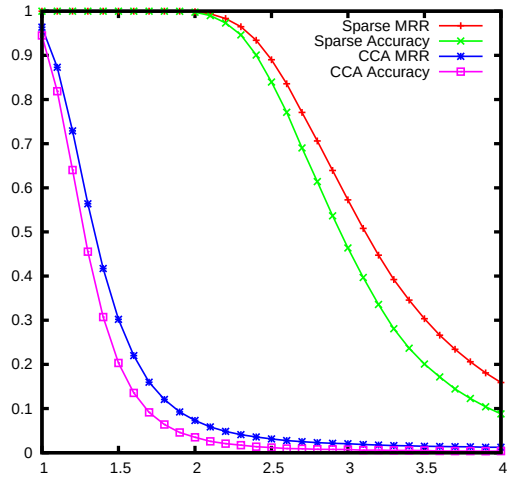


Figure 1: Accuracy of CCA and our sparsified version with the noise parameter.

We generate a total of 3000 pairs of points and use 2000 of them for training the models and the rest for evaluation. We use the true feature correspondences to form the cross-covariance selection matrix I^{xy} (Sec. 3.3). For this experiment, we use the full monolingual covariance matrices. We train both CCA and our sparse version on the training data and evaluate them on the test data. We repeat this multiple times and report the average accuracies. Fig. 1 shows the performance of CCA and our sparse CCA, as we vary the noise parameter η from 1 to 4. It is very clear that the sparse version performs significantly better than CCA. As the noise increases, the performance of CCA drops quickly. This experiment demonstrates a significant performance gain when the true correspondences are available. But this information is not available in the case of real world data sets, so we try to approximate it.

4.3 Model Selection

As we have discussed, there are several choices for computing the association measure and for selecting the word pairs to be retained. And each of them have sparsity parameters, giving raise to many possible models. For model selection, we use approximately 5000 document pairs collected from the Wikipedia between English and Spanish. We use the cross-language links provided as the ground truth. We tokenize the documents, retain only the most frequent 2000 words in each language and convert the docu-

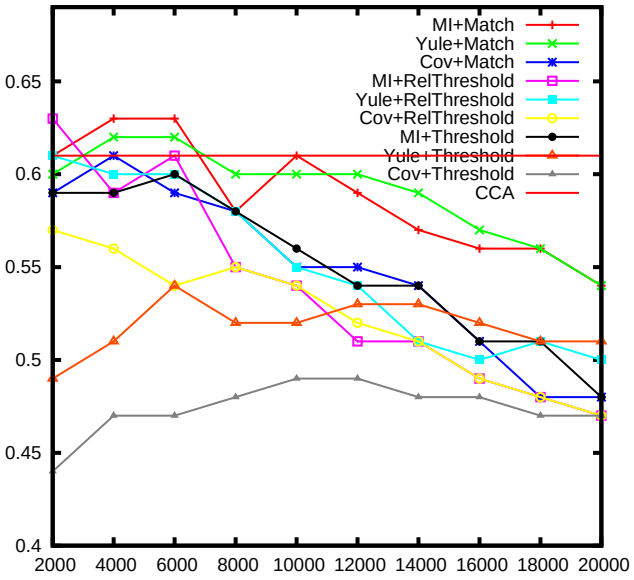


Figure 2: Comparison of the word association measures along with different selection criteria. The x-axis plots the number of non-zero entries in the covariance matrices and the y-axis plots the accuracy of top-ranked document.

ments into TFIDF vectors. We use 60% of the data for training different models and the rest for evaluating the models. We choose a few promising models based on this development set results and evaluate them on bigger data sets.

4.3.1 Selection Strategies

In the first experiment, we combine the three association measures, Covariance (Cov), MI and Yule's ω, with the three selection criteria, Threshold, Relative Threshold (RelThreshold) and Matching (Match). Fig. 2 shows the performance of these different combinations with varying levels of sparsity in the covariance matrices. The horizontal line represents the performance of CCA on this data set. We start with 2000 non-zero entries in the covariance matrices and experiment up to 20,000 non-zero entries. Since our data set has 2000 words in each language, 2000 non-zero entries in a covariance matrix implies that, on an average, every word is associated with only one word. This results in highly sparse covariance matrices.

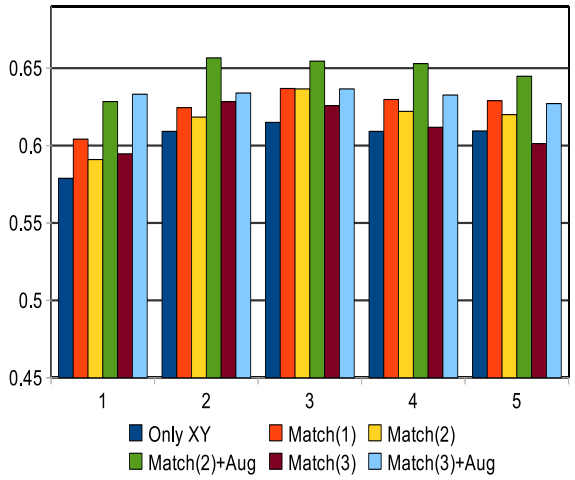
Overall, we observe that reducing the level of sparsity , *i.e.* selecting more number of elements in the covariance matrices, increases the performance slightly and then decreases again. From the figure, it seems that sparsifying the covariance matrices might help in improving the performance of the task. But it is interesting to note that not all the models perform better than CCA. In fact, both the models that achieve better scores use Matching as the selection criteria. This suggests that, apart from the weighting of the word pairs, appropriate selection of the word pairs is also equally important. In the rest of the experiments we mainly report results with Matching as the selection criterion. From this figure, we observe that Mutual Information and Yule's ω perform competitively but they consistently outperform models that use covariance as the association measure. So in the rest of the experiments we report results with MI or Yule's ω.

4.3.2 Amount of Sparsity

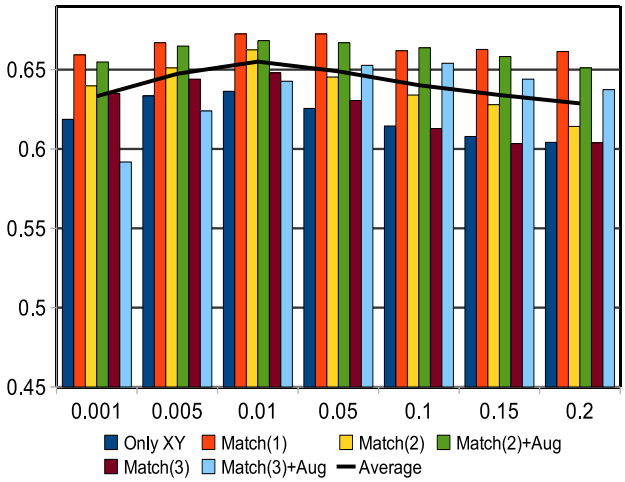
In the previous experiment, we used same level of sparsity for all the covariance matrices, *i.e.* same number of associations were selected for each word in all the three covariance matrices. In the following experiment, we use different levels of sparsity for the individual covariance matrices. Fig. 3 shows the performance of Yule+Match and Dictionary+Match combinations with different levels of sparsity. In the Yule+Match combination, we use Yule's ω association measure for weighting the word pairs and use matching for selection. In the Dictionary+Match combination, we use bilingual dictionary for sparsifying cross-covariance matrix, *i.e.* we keep all the word pairs whose conditional translation probability is above a threshold. And for monolingual word pairs, we use MI for weighting and matching for word pair selection.

For each level of sparsity of the cross-covariance matrix, we experiment with different levels of sparsity on the monolingual covariance matrices. 'Only XY' indicates we use the full monolingual covariance matrices. In 'Match(k)' runs, we allow each word to be associated with a total of k words (Eq. 5). 'Aug' indicates that we use monolingual augmentation to refine the sparsity of the cross-covariance matrix (Sec. 3.2.4).

From both the figures 3(a) and 3(b), we observe that 'Only XY' run (dark blue) performs poorly compared to the other runs, indicating that sparsifying all the covariance matrices is better than sparsifying only the cross-covariance matrix. In the



(a) Performance of Yule+Match combination. The x-axis plots the number of Spanish words selected per each English word and vice versa. This determines the sparsity of C^{xy}. Matching is used as selection criteria for all the covariance matrices.



(b) Performance of Dictionary+Match combination. The x-axis plots the threshold on bilingual translation probability and it determines the sparsity of C^{xy}. Matching is used to select *only* the monolingual sparsity.

Figure 3: Comparison of Yule+Match and Dictionary+Match combination with different levels of sparsity for the covariance matrices. In both the figures, the x-axis plots the sparsity of the cross-covariance matrix and for each value we try different levels of sparsity on the monolingual covariance matrices (which are grouped together). The description of these individual runs is provided in the relevant parts of the text. The y-axis plots the accuracy of the top-ranked document. CCA achieves 61% accuracy on this data set.

Yule+Match combination, Fig. 3(a), all the runs seem to be performing better when each English word is allowed to associate with 2 or 3 Spanish words and vice versa. Among different ways of selecting the monolingual word pairs, Match(2)+Aug performs better than the remaining runs. So we use Match(2)+Aug combination for the Yule's ω measure.

Unlike the Yule+Match combinations, there is no clear winner for Dictionary+Match combinations. First of all, the performance increase as we increase the translation probability threshold and then decreases again (indicated by the 'Average' performance in Fig. 3(b)). On an average, all the systems perform better with a threshold of 0.01, which we use in our final experiments. In this case, both Match(1) and Match(2)+Aug runs (orange and green bars respectively) perform competitively so we use both of these models in our final experiments.

In both the above experiments, the performance bars are very similar when we use MI instead of Yule and vice versa for weighting monolingual word pairs. Thus, to illustrate the main ideas we chose Yule's ω for the former combination and MI for the latter combination.

4.3.3 Promising Models

Based on the above experiments, we choose the following combinations for our final experiments. Yule(l)+Match(k), where $l \in \{2, 3\}$ is the number of Spanish words allowed for each English word and vice versa and k=2 is the number of monolingual word associations for each word. We also run both these combinations with monolingual augmentation, indicated by Yule(l)+Match(k)+Aug. For dictionary based weighting, Dictionary+Match(k), we choose a translation probability threshold of 0.01 and try $k \in \{1, 2\}$. Again, we run these combinations with monolingual augmentation identified by Dictionary+Match(k)+Aug.

4.4 Results

For our final results, we choose data in two language pairs (English-Spanish and English-German) from two different resources, Europarl (Koehn, 2005) and Wikipedia. For Europarl data sets, we artificially make them comparable by considering the first half

	Wikipedia				Europarl			
	English-Spanish		English-German		English-Spanish		English-German	
	Acc.	MRR	Acc.	MRR	Acc.	MRR	Acc.	MRR
CCA	0.776	0.852	0.570	0.699	0.872	0.920	0.748	0.831
OPCA	0.781	0.856	0.570	0.700	0.870	0.920	0.748	0.831
Yule(2)+Match(2)	0.798*	0.866*	0.576	0.703	0.901*	0.939*	0.780*	0.853*
Yule(2)+Match(2)+Aug	0.811*	0.876*	0.602*	0.723*	0.883	0.927	0.771*	0.847*
Yule(3)+Match(2)	0.803*	0.870*	0.572	0.700	0.856	0.907	0.747	0.830
Yule(3)+Match(2)+Aug	0.793*	0.861*	0.610*	0.726*	0.878$^+$	0.925$^+$	0.763$^+$	0.843*
Dictionary+Match(1)	0.811*	0.875*	**0.656***	**0.762***	**0.928***	**0.957***	**0.874***	**0.922***
Dictionary+Match(2)	0.811*	0.876*	0.623*	0.736*	0.923*	0.955*	0.853*	0.907*
Dictionary+Match(2)+Aug	**0.825***	**0.885***	0.630*	0.735*	0.897*	0.935*	0.866*	0.917*

Table 1: Performance of our models in comparison with CCA and OPCA on English-Spanish and English-German language pairs. * and $^+$ indicate statistical significance measured by paired t-test at p=0.01 and 0.05 levels respectively. When an improvement is significant at p=0.01 it is automatically significant at p=0.05 and hence is not shown.

of English document and the second half of its aligned foreign language document (Mimno et al., 2009). For Wikipedia data set, we use the cross-language link as the ground truth. For each of these data sets, we choose approximately 5000 aligned document pairs. We remove the stop words and keep all the words that occur in at least five documents. After the preprocessing, on an average, we are left with 4700 words in each language. Subsequently we convert the documents into their TFIDF representation.

In Platt *et al.* (2010), the authors compare different systems on the comparable document retrieval task and show that discriminative approaches work better compared to their generative counter parts. So, here we compare only with the state-of-the-art discriminative systems such as CCA and OPCA (Platt et al., 2010). For each of the systems, we report the average results of five-fold cross validation. We divide the data into 3:1:1 ratio for training, validation and test sets. The validation data set is used to select the best number of dimensions of the common sub space. For both CCA and our models, we set the regularization parameter λ to 0.3 which we found works well in a relevant but different experiments. For OPCA, we manually tried different regularization parameters ranging from 0.0001 to 1 and found that a value of 0.001 worked best.

The results are shown in Table 1. On these data sets, both CCA and OPCA performed competitively. OPCA takes advantage of the common vocabulary in both the languages. But in our data sets, vocabulary of both the languages is treated differently, so it is not surprising that they give almost the same results. From the results, it is clear that sparsifying the covariance matrices helps improving the accuracies significantly. In all the four data sets, the best performing method always used dictionary for cross-lingual sparsity selection. This indicates that using fine granular information such as a bilingual dictionary gleaned from an external source is very helpful in improving the accuracies. Among the models that rely solely on the training data, models that use monolingual augmentation performed better on Wikipedia data set, while models that do not use augmentation performed better on Europarl data sets. This suggests that, when the aligned documents are clean (closer to being parallel) the statistics computed from cross-lingual corpora are trustworthy. As the documents become comparable, we need to use monolingual statistics to refine the bilingual statistics. Moreover, these models achieve higher gains in the case of Wikipedia data set compared to the gains in Europarl. This conforms with our initial hunch that, when the training data is clean the covariance matrices tend to be less noisy.

5 Discussion

In this paper, we have proposed the idea of sparsifyng covariance matrices to improve bilingual pro-

jection directions. We are not aware of any NLP research that attempts to recover the sparseness of the covariance matrices to improve the projection directions. Our work is different from the sparse CCA (Hardoon and Shawe-Taylor, 2011; Rai and Daumé III, 2009) proposed in the Machine Learning literature. Their objective is to find projection directions such that the original documents are represented as a sparse vectors in the common sub-space. Another seemingly relevant but different direction is the sparse covariance matrix selection research (Banerjee et al., 2005). The objective in this work is to find matrices such that the *inverse* of the covariance matrix is sparse which has applications in Gaussian processes.

In this paper, we tried sparsification in the context of CCA only but our technique is general and can be applied to its variants like OPCA. Our experimental results show that using external information such as bilingual dictionaries which is gleaned from cleaner resources brings significant improvements. Moreover, we also observe that computing word pair association measures from the same training data along with an appropriate selection criteria can *also* yield significant improvements. This is certainly encouraging and in future we would like to explore more sophisticated techniques to recover the sparsity based on the training data itself.

6 Acknowledgments

We thank the anonymous reviewers for their helpful comments. This material is partially supported by the National Science Foundation under Grant No. 1139909.

References

Francis R. Bach and Michael I. Jordan. 2005. A probabilistic interpretation of canonical correlation analysis. Technical report, Dept Statist Univ California Berkeley CA Tech.

Lisa Ballesteros and W. Bruce Croft. 1996. Dictionary methods for cross-lingual information retrieval. In *Proceedings of the 7th International Conference on Database and Expert Systems Applications*, DEXA '96, pages 791–801, London, UK. Springer-Verlag.

Onureena Banerjee, Alexandre d'Aspremont, and Laurent El Ghaoui. 2005. Sparse covariance selection via robust maximum likelihood estimation. *CoRR*, abs/cs/0506023.

Nuria Bel, Cornelis H. A. Koster, and Marta Villegas. 2003. Cross-lingual text categorization.

Hal Daume III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 407–412, Portland, Oregon, USA, June. Association for Computational Linguistics.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, March.

William A. Gale and Kenneth W. Church. 1991. A program for aligning sentences in bilingual corpora. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 177–184, Morristown, NJ, USA. Association for Computational Linguistics.

Wei Gao, John Blitzer, Ming Zhou, and Kam-Fai Wong. 2009. Exploiting bilingual information to improve web search. In *Proceedings of Human Language Technologies: The 2009 Conference of the Association for Computational Linguistics*, ACL-IJCNLP '09, pages 1075–1083, Morristown, NJ, USA. ACL.

Aria Haghighi, Percy Liang, Taylor B. Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*, pages 771–779, Columbus, Ohio, June. Association for Computational Linguistics.

David R. Hardoon and John Shawe-Taylor. 2011. Sparse canonical correlation analysis. *Journal of Machine Learning*, 83(3):331–353.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proceedings of ACL-08: HLT*, pages 389–397, Columbus, Ohio, June. Association for Computational Linguistics.

Hung Huu Hoang, Su Nam Kim, and Min-Yen Kan. 2009. A Re-examination of Lexical Association Measures. In *Proceedings of ACL-IJCNLP 2009 Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, Singapore, August. Association for Computational Linguistics.

H. Hotelling. 1936. Relation between two sets of variables. *Biometrica*, 28:322–377.

Diana Zaiu Inkpen and Graeme Hirst. 2002. Acquiring collocations for lexical choice between near-synonyms. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition - Volume 9*, ULA '02, pages 67–76, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jagadeesh Jagarlamudi and Hal Daumé III. 2010. Extracting multilingual topics from unaligned comparable corpora. In *Advances in Information Retrieval, 32nd European Conference on IR Research, ECIR*, volume 5993, pages 444–456, Milton Keynes, UK. Springer.

Jagadeesh Jagarlamudi, Hal Daume III, and Raghavendra Udupa. 2011. From bilingual dictionaries to interlingual document representations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 147–152, Portland, Oregon, USA, June. Association for Computational Linguistics.

R. Jonker and A. Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 817–824, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 880–889, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert C. Moore. 2004. On Log-Likelihood-Ratios and the Significance of Rare Events. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 333–340, Barcelona, Spain, July. Association for Computational Linguistics.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Comput. Linguist.*, 31:477–504, December.

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

John C. Platt, Kristina Toutanova, and Wen-tau Yih. 2010. Translingual document representations from discriminative projections. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 251–261, Stroudsburg, PA, USA. Association for Computational Linguistics.

Piyush Rai and Hal Daumé III. 2009. Multi-label prediction via sparse infinite cca. In *Advances in Neural Information Processing Systems*, Vancouver, Canada.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 519–526, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2009. Learning phoneme mappings for transliteration without parallel data. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 37–45, Boulder, Colorado, June. Association for Computational Linguistics.

K. Ahuja Ravindra, L. Magnanti Thomas, and B. Orlin James. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc.

Harry T Reis and Charles M Judd. 2000. *Handbook of Research Methods in Social and Personality Psychology*. Cambridge University Press.

Alexander Schrijver. 2003. *Combinatorial Optimization*. Springer.

Michael L. Littman Susan T. Dumais, Thomas K. Landauer. 1996. Automatic cross-linguistic information retrieval using latent semantic indexing. In *Working Notes of the Workshop on Cross-Linguistic Information Retrieval, SIGIR*, pages 16–23, Zurich, Switzerland. ACM.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res. (JAIR)*, 37:141–188.

Alexei Vinokourov, John Shawe-taylor, and Nello Cristianini. 2003. Inferring a semantic representation of text via cross-language correlation analysis. In *Advances in Neural Information Processing Systems*, pages 1473–1480, Cambridge, MA. MIT Press.

Thuy Vu, AiTi Aw, and Min Zhang. 2009. Feature-based method for document alignment in comparable news corpora. In *EACL*, pages 843–851.

Duo Zhang, Qiaozhu Mei, and ChengXiang Zhai. 2010. Cross-lingual latent topic extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1128–1137, Uppsala, Sweden, July. Association for Computational Linguistics.

Entire Relaxation Path for Maximum Entropy Problems

Moshe Dubiner
Google
moshe@google.com

Yoram Singer
Google
singer@google.com

Abstract

We discuss and analyze the problem of finding a distribution that minimizes the relative entropy to a prior distribution while satisfying max-norm constraints with respect to an observed distribution. This setting generalizes the classical maximum entropy problems as it relaxes the standard constraints on the observed values. We tackle the problem by introducing a re-parametrization in which the unknown distribution is distilled to a single scalar. We then describe a homotopy between the relaxation parameter and the distribution characterizing parameter. The homotopy also reveals an aesthetic symmetry between the prior distribution and the observed distribution. We then use the reformulated problem to describe a space and time efficient algorithm for tracking the *entire* relaxation path. Our derivations are based on a compact geometric view of the relaxation path as a piecewise linear function in a *two* dimensional space of the relaxation-characterization parameters. We demonstrate the usability of our approach by applying the problem to Zipfian distributions over a large alphabet.

1 Introduction

Maximum entropy (max-ent) models and its dual counterpart, logistic regression, is a popular and effective tool in numerous natural language processing tasks. The principle of maximum entropy was spelled out explicitly by E.T. Jaynes (1968). Applications of maximum entropy approach to natural language processing are numerous. A notable example and probably one of the earliest usages and generalizations of the maximum entropy principle to language processing is the work of Berger, Della Pietra$\times 2$, and Lafferty (Berger et al., 1996, Della Pietra et al., 1997). The original formulation of max-ent cast the problem as the task of finding the distribution attaining the highest entropy subject to *equality* constraints. While this formalism is aesthetic and paves the way to a simple dual in the form of a unique Gibbs distribution (Della Pietra et al., 1997), it does not provide sufficient tools to deal with input noise and sparse representation of the target Gibbs distribution. To mitigate these issues, numerous relaxation schemes of the equality constraints have been proposed. A notable recent work by Dudik, Phillips, and Schapire (2007) provided a general constraint-relaxation framework. See also the references therein for an in depth overview of other approaches and generalizations of max-ent. The constraint relaxation surfaces a natural parameter, namely, a relaxation value. The dual form of this free parameter is the regularization value of penalized logistic regression problems. Typically this parameter is set by experimentation using cross validation technique. The relaxed maximum-entropy problem setting is the starting point of this paper.

In this paper we describe and analyze a framework for *efficiently* tracking the entire relaxation path of constrained max-ent problems. We start in Sec. 2 with a generalization in which we discuss the problem of finding a distribution that minimizes the relative entropy to a given prior distribution while satisfying max-norm constraints with respect to an observed distribution. In Sec. 3 we tackle the problem by introducing a re-parametrization in which the

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 941–948,
Edinburgh, Scotland, UK, July 27–31, 2011.

unknown distribution is distilled to a single scalar. We next describe in Sec. 4 a homotopy between the relaxation parameter and the distribution characterizing parameter. This formulation also reveals an aesthetic symmetry between the prior distribution and the observed distribution. We use the reformulated problem to describe in Secs. 5-6 space and time efficient algorithms for tracking the *entire* relaxation path. Our derivations are based on a compact geometric view of the relaxation path as a piecewise linear function in a *two* dimensional space of the relaxation-characterization parameters. In contrast to common homotopy methods for the Lasso Osborne et al. (2000), our procedure for tracking the max-ent homotopy results in an uncharacteristically low complexity bounds thus renders the approach applicable for large alphabets. We provide preliminary experimental results with Zipf distributions in Sec. 8 that demonstrate the merits of our approach. Finally, we conclude in Sec. 9 with a brief discussion of future directions.

2 Notations and Problem Setting

We denote vectors with bold face letters, e.g. $\boldsymbol{v}$. Sums are denoted by calligraphic letters, e.g. $\mathcal{M} = \sum_j m_j$. We use the shorthand $[n]$ to denote the set of integers $\{1, \ldots, n\}$. The n'th dimensional simplex, denoted Δ, consists of all vectors $\boldsymbol{p}$ such that, $\sum_{j=1}^n p_j = 1$ and for all $j \in [n]$, $p_j \geq 0$. We generalize this notion to multiplicity weighted vectors. Formally, we say that a vector $\boldsymbol{p}$ with multiplicity $\boldsymbol{m}$ is in the simplex, $(\boldsymbol{p}, \boldsymbol{m}) \in \Delta$, if $\sum_{j=1}^n m_j p_j = 1$, and for all $j \in [n]$, $p_j \geq 0$, and $m_j \geq 0$.

The generalized relaxed maximum-entropy problem is concerned with obtaining an estimate $\boldsymbol{p}$, given a prior distribution $\boldsymbol{u}$ and an observed distribution $\boldsymbol{q}$ such that the relative entropy between $\boldsymbol{p}$ and $\boldsymbol{u}$ is as small as possible while $\boldsymbol{p}$ and $\boldsymbol{q}$ are within a given max-norm tolerance. Formally, we cast the following constrained optimization problem,

$$\min_{\boldsymbol{p}} \sum_{j=1}^n m_j p_j \log\left(\frac{p_j}{u_j}\right) \quad , \tag{1}$$

such that $(\boldsymbol{p}, \boldsymbol{m}) \in \Delta$; $\|\boldsymbol{p} - \boldsymbol{q}\|_\infty \leq 1/\nu$. The vectors $\boldsymbol{u}$ and $\boldsymbol{q}$ are dimensionally compatible with $\boldsymbol{p}$, namely, $(\boldsymbol{q}, \boldsymbol{m}) \in \Delta$ and $(\boldsymbol{u}, \boldsymbol{m}) \in \Delta$. The scalar ν is a relaxation parameter. We use $1/\nu$ rather than ν itself for reasons that become clear in the sequel.

We next describe the dual form of (1). We derive the dual by introducing Lagrange-Legendre multipliers for each of the constraints appearing in (1). Let $\alpha_j^+ \geq 0$ denote the multiplier for the constraint $q_j - p_j \leq 1/\nu$ and $\alpha_j^- \geq 0$ the multiplier for the constraint $q_j - p_j \geq -1/\nu$. In addition, we use γ as the multiplier for the constraint $\sum_j m_j p_j = 1$. fter some routine algebraic manipulations we get that the Lagrangian is,

$$\begin{aligned} &\textstyle\sum_{j=1}^n m_i \left(p_j \log\left(\frac{p_j}{u_j}\right) + \alpha_j (q_j - p_j) + \frac{|\alpha_j|}{\nu} \right) \\ &\qquad + \gamma \left(\textstyle\sum_{j=1}^n m_j p_j - 1 \right) \quad . \end{aligned} \tag{2}$$

To find the dual form we take the partial derivative of the Lagrangian with respect to each p_j, equate to zero, and get that $\log\left(\frac{p_j}{u_j}\right) + 1 - \alpha_j + \gamma = 0$, which implies that $p_j \sim u_j e^{\alpha_j}$. We now employ the fact that $(\boldsymbol{p}, \boldsymbol{m}) \in \Delta$ to get that the exact form for p_j is

$$p_j = \frac{u_j \mathrm{e}^{\alpha_j}}{\sum_{i=1}^n m_i u_i \mathrm{e}^{\alpha_i}} \quad . \tag{3}$$

Using (3) in the compact form of the Lagrangian we obtain the following dual problem

$$\max_{\boldsymbol{\alpha}} - \left\{ \log(Z) - \sum_{j=1}^n m_j q_j \alpha_j + \sum_{j=1}^n \frac{m_j}{\nu} |\alpha_j| \right\} \quad , \tag{4}$$

where $Z = \sum_{j=1}^n m_j u_j \mathrm{e}^{\alpha_j}$. We make rather little use of the dual form of the problem. However, the complementary slackness conditions that are necessary for optimality to hold play an important role in the next section in which we present a reformulation of the relaxed maximum entropy problem.

3 Problem Reformulation

First note that the primal problem is a strictly convex function over a compact convex domain. Thus, its optimum exists and is unique. Let us now characterize the form of the solution. We partition the set of indices in $[n]$ into three disjoint sets depending on whether the constraint $|p_j - q_j| \leq 1/\nu$ is active and its form. Concretely, we define

$$\begin{aligned} I_- &= \{1 \leq j \leq n \mid p_j = q_j - 1/\nu\} \\ I_0 &= \{1 \leq j \leq n \mid |p_j - q_j| < 1/\nu\} \\ I_+ &= \{1 \leq j \leq n \mid p_j = q_j + 1/\nu\} \quad . \end{aligned} \tag{5}$$

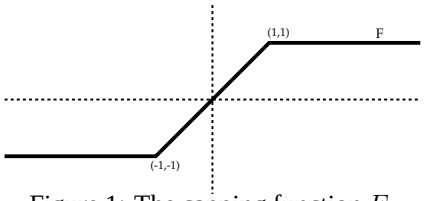


Figure 1: The capping function F.

Recall that $Z = \sum_{j=1}^{n} m_j u_j e^{\alpha_j}$. Thus, from (3) we can rewrite $p_j = u_j e^{\alpha_j}/Z$. We next use the complementary slackness conditions (see for instance (Boyd and Vandenberghe, 2004)) to further characterize the solution. For any $j \in I_-$ we must have $\alpha_j^- = 0$ and $\alpha_j^+ \geq 0$ therefore $\alpha_j \geq 0$, which immediately implies that $p_j \geq u_j/Z$. By definition we have that $p_j = q_j - 1/\nu$ for $j \in I_-$. Combining these two facts we get that $u_j/Z \leq q_j - 1/\nu$ for $j \in I_-$. Analogous derivation yields that $u_j/Z \geq q_j + 1/\nu$ for $j \in I_+$. Last, if the set I_0 is not empty then for each j in I_0 we must have $\alpha_j^+ = 0$ and $\alpha_j^- = 0$ thus $\alpha_j = 0$. Resorting again to the definition of $\boldsymbol{p}$ from (3) we get that $p_j = u_j/Z$ for $j \in I_0$. Since $|p_j - q_j| < 1/\nu$ for $j \in I_0$ we get that $|u_j/Z - q_j| < 1/\nu$. To recap, there exists $Z > 0$ such that the optimal solution takes the following form,

$$p_j = \begin{cases} q_j - 1/\nu & u_j/Z \leq q_j - 1/\nu \\ u_j/Z & |u_j/Z - q_j| < 1/\nu \\ q_j + 1/\nu & u_j/Z \geq q_j + 1/\nu \end{cases} \quad . \tag{6}$$

We next introduce an key re-parametrization, defining $\mu = \nu/Z$. We also denote by $F(\cdot)$ the capping function $F(x) = \max\{-1, \min\{1, x\}\}$. A simple illustration of the capping function is given in Fig. 1. Equipped with these definition we can rewrite (6) as follows,

$$p_j = q_j + \frac{1}{\nu} F(\mu u_j - \nu q_j) \ . \tag{7}$$

Given $\boldsymbol{u}$, $\boldsymbol{q}$, and ν, the value of μ can be found by using $\sum_j m_j p_j = \sum_j m_j q_j = 1$, which implies

$$G(\nu, \mu) \stackrel{\text{def}}{=} \sum_{j=1}^{n} m_j F(\mu u_j - \nu q_j) = 0 \ . \tag{8}$$

We defer the derivation of the actual algorithm for computing μ (and in turn $\boldsymbol{p}$) to the next section. In the meanwhile let us continue to explore the rich structure of the general solution. Note that $\mu, \boldsymbol{u}$ are interchangeable with $\nu, \boldsymbol{q}$. We can thus swap the roles of the prior distribution with the observed distribution and obtain an analogous characterization. In the next section we further explore the dependence of μ on ν. The structure we reveal shortly serves as our infrastructure for deriving efficient algorithms for following the regularization path.

4 The function $\mu(\nu)$

In order to explore the dependency of μ on ν let us introduce the following sums

$$\begin{aligned} \mathcal{M} &= \sum_{j \in I_+} m_j - \sum_{j \in I_-} m_j \\ \mathcal{U} &= \sum_{j \in I_0} m_j \, u_j \\ \mathcal{Q} &= \sum_{j \in I_0} m_j \, q_j \ . \end{aligned} \tag{9}$$

Fixing ν and using (9), we can rewrite (8) as follows

$$\mu \mathcal{U} - \nu \mathcal{Q} + \mathcal{M} = 0 \ . \tag{10}$$

Clearly, so long as the partition of $[n]$ into the sets I_+, I_-, I_0 is intact, there is a simple linear relation between μ and ν. The number of possible subsets I_-, I_0, I_+ is finite. Thus, the range $0 < \nu < \infty$ decomposes into a finite number of intervals each of which corresponds to a fixed partition of $[n]$ into I_+, I_-, I_0. In each interval μ is a linear function of ν, unless I_0 is empty. Let ν_∞ be the smallest ν value for which I_0 is empty. Let μ_∞ be its corresponding μ value. If I_0 is never empty for any finite value of ν we define $\nu_\infty = \mu_\infty = \infty$. Clearly, replacing (ν, μ) with $(\kappa\nu, \kappa\mu)$ for any $\kappa \geq 1$ and $\nu \geq \nu_\infty$ yields the same feasible solution as $I_+(\kappa\nu) = I_+(\nu)$, $I_-(\alpha\nu) = I_-(\nu)$. Hence, as far as the original problem is concerned there is no reason to go past ν_∞ during the process of characterizing the solution. We recap our derivation so far in the following lemma.

Lemma 4.1 *For $0 \leq \nu \leq \nu_\infty$, the value of μ as defined by (7) is a unique. Further, the function $\mu(\nu)$ is a piecewise linear continuous function in ν. When $\nu \geq \nu_\infty$ letting $\mu = \mu_\infty \nu/\nu_\infty$ keeps (7) valid.*

We established the fact that $\mu(\nu)$ is a piecewise linear function. The lingering question is how many

linear sub-intervals the function can attain. To study this property, we take a geometric view of the plane defined by (ν, μ). Our combinatorial characterization of the number of sub-intervals makes use of the following definitions of lines in $\mathbb{R}^2$,

$$\ell_{+j} = \{(\nu,\mu) \mid u_j\mu - q_j\nu = +1\} \quad (11)$$
$$\ell_{-j} = \{(\nu,\mu) \mid u_j\mu - q_j\nu = -1\} \quad (12)$$
$$\ell_0 = \{(\nu,\mu) \mid \mu\mathcal{U} - \nu\mathcal{Q} + \mathcal{M} = 0\} \ , \quad (13)$$

where $-\infty < \nu < \infty$ and $j \in [n]$. The next theorem gives an upper bound on the number of linear segments the function $\mu()$ may attain. While the bound is quadratic in the dimension, for both artificial data and real data the bound is way too pessimistic.

Theorem 4.2 *The piecewise linear function $\mu(\nu)$ consists of at most n^2 linear segments for $\nu \in \mathbb{R}_+$.*

Proof Since we showed that that $\mu(\nu)$ is a piecewise linear function, it remains to show that it has at most n^2 linear segments. Consider the two dimensional function $G(\nu, \mu)$ from (8). The (ν, μ) plane is divided by the $2n$ straight lines $\ell_1, \ell_2, \ldots, \ell_n, \ell_{-1}, \ell_{-2}, \ldots, \ell_{-n}$ into at most $2n^2+1$ polygons. The latter property is proved by induction. It clearly holds for $n = 0$. Assume that it holds for $n-1$. Line ℓ_n intersects the previous $2n-2$ lines at no more than $2n-2$ points, thus splitting at most $2n-1$ polygons into two separate polygonal parts. Line ℓ_{-n} is parallel to ℓ_n, again adding at most $2n-1$ polygons. Recapping, we obtain at most $2(n-1)^2 + 1 + 2(2n-1) = 2n^2 + 1$ polygons, as required per induction. Recall that $\mu(\nu)$ is linear inside each polygon. The two extreme polygons where $G(\nu, \mu) = \pm\sum_{j=1}^n m_j$ clearly disallow $G(\nu, \mu) = 0$, hence $\mu(\nu)$ can have at most $2n^2 - 1$ segments for $-\infty < \nu < \infty$. Lastly, we use the symmetry $G(-\nu, -\mu) = -G(\nu, \mu)$ which implies that for $\nu \in \mathbb{R}_+$ there are at most n^2 segments. ■

This result stands in contrast to the Lasso homotopy tracking procedure (Osborne et al., 2000), where the worst case number of segments seems to be exponential in n. Moreover, when the prior $\boldsymbol{u}$ is uniform, $u_j = 1/\sum_{j=1}^n m_j$ for all $j \in [n]$, the number of segments is at most $n+1$. We defer the analysis of the uniform case to a later section as the proof stems from the algorithm we describe in the sequel.

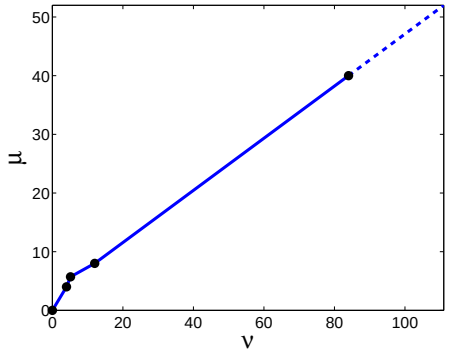


Figure 2: An illustration of the function $\mu(\nu)$ for a synthetic 3 dimensional example.

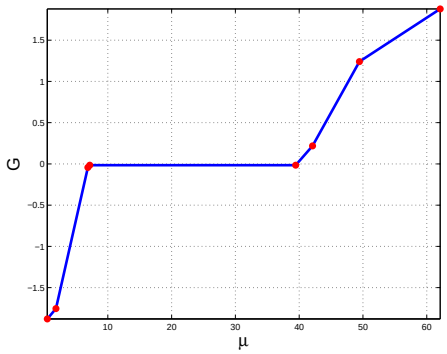


Figure 3: An illustration of the function $G(\mu)$ for a synthetic 4 dimensional example and a $\nu = 17$.

5 Algorithm for a Single Relaxation Value

Suppose we are given $\boldsymbol{u}, \boldsymbol{q}, \boldsymbol{m}$ and a specific relaxation value $\tilde{\nu}$. How can we find $\boldsymbol{p}$? The obvious approach is to solve the one dimensional monotonically nondecreasing equation $G(\mu) \stackrel{\text{def}}{=} G(\tilde{\nu}, \mu) = 0$ by bisection. In this section we present a more efficient and direct procedure that is guaranteed to find the optimal solution $\boldsymbol{p}$ in a finite number of steps. Clearly $G(\mu)$ is a piecewise linear function with at most $2n$ easily computable change points of the slope. See also Fig. (5) for an illustration of $G(\cdot)$. In order to find the slope change points we need to calculate the point (ν, μ_j) for all the lines $\ell_{\pm j}$ where $1 \le j \le n$. Concretely, these values are

$$\mu_j = \frac{\nu q_{|j|} + \text{sign}(j)}{u_{|j|}} \ . \quad (14)$$

We next sort the above values of μ_j and denote the resulting sorted list as $\mu_{\pi_1} \le \mu_{\pi_2} \le \cdots \le \mu_{\pi_{2n}}$. For any $0 \le j \le 2n$ let $\mathcal{M}_j, \mathcal{U}_j, \mathcal{Q}_j$ be the sums, defined

in (9), for the line segment $\mu_{\pi_{j-1}} < \mu < \mu_{\pi_j}$ (denoting $\mu_{\pi_0} = -\infty$, $\mu_{\pi_{2n+1}} = \infty$). We compute the sums $\mathcal{M}_j, \mathcal{U}_j, \mathcal{Q}_j$ incrementally, starting from $\mathcal{M}_0 = -\sum_{i=1}^n m_i$, $\mathcal{U}_0 = \mathcal{Q}_0 = 0$. Once the values of $j-1$'th sums are known, we can compute the next sums in the sequence as follows,

$$\begin{aligned} \mathcal{M}_j &= \mathcal{M}_{j-1} + m_{|\pi_j|} \\ \mathcal{U}_j &= \mathcal{U}_{j-1} - \mathrm{sign}(\pi_j)\, m_{|\pi_j|}\, u_{|\pi_j|} \\ \mathcal{Q}_j &= \mathcal{Q}_{j-1} - \mathrm{sign}(\pi_j)\, m_{|\pi_j|}\, q_{|\pi_j|} \ . \end{aligned}$$

From the above sums we can compute the value of the function $G(\nu, \mu)$ at the end point of the line segment $(\mu_{\pi_{j-1}}, \mu_{\pi_j})$, which is the same as the start point of the line segment $(\mu_{\pi_j}, \mu_{\pi_{j+1}})$,

$$\begin{aligned} G_j &= \mathcal{M}_{j-1} + \mathcal{U}_{j-1}\,\mu_j - \mathcal{Q}_{j-1}\,\nu \\ &= \mathcal{M}_j + \mathcal{U}_j\,\mu_j - \mathcal{Q}_j\,\nu \ . \end{aligned}$$

The optimal value of μ resides in the line segment for which $G(\cdot)$ attains 0. Such a segment must exist since $G_0 = \mathcal{M}_0 = -\sum_{i=1}^n m_i < 0$ and $G_{2n} = -\mathcal{M}_0 > 0$. Therefore, there exists an index $1 \le j < 2n$, where $G_j \le 0 \le G_{j+1}$. Once we bracketed the feasible segment for μ, the optimal value of μ is found by solving the linear equation (10),

$$\mu = \left(\mathcal{Q}_j\,\nu - \mathcal{M}_j\right)/\mathcal{U}_j \quad . \tag{15}$$

From the optimal value of μ it is straightforward to construct $\boldsymbol{p}$ using (7). Due to the sorting step, the algorithm's run time is $O(n\log(n))$ and it takes linear space. The number of operations can be reduced to $O(n)$ using a randomized search procedure.

6 Homotopy Tracking

We now shift gears and focus on the main thrust of this paper, namely, an efficient characterization of the *entire* regularization path for the maximum entropy problem. Since we have shown that the optimal solution $\boldsymbol{p}$ can be straightforwardly obtained from the variable μ, it suffices to efficiently track the function $\mu(\nu)$ as we traverse the plane (ν, μ) from $\nu = 0$ through the last change point which we denoted as (ν_∞, μ_∞). In this section we give an algorithm that traverses $\mu(\nu)$ by locating the intersections of ℓ_0 with the fixed lines $\ell_{-n}, \ell_{-n+1}, \ldots, \ell_{-1}, \ell_1, \ldots, \ell_n$ and updating ℓ_0 after each intersection.

More formally, the local homotopy tracking follows the piecewise linear function $\mu(\nu)$, segment by segment. Each segment corresponds to a subset of the line ℓ_0 for a *given* triplet $(\mathcal{M}, \mathcal{U}, \mathcal{Q})$. It is simple to show that $\mu(0) = 0$, hence we start with

$$\nu = 0,\ \mathcal{M} = 0,\ \mathcal{U} = \mathcal{Q} = 1 \ . \tag{16}$$

We now track the value of μ as ν increases, and the relaxation parameter $1/\nu$ decreases. The characterization of ℓ_0 remains intact until ℓ_0 hits one of the lines ℓ_j for $1 \le |j| \le n$. To find the line intersecting ℓ_0 we need to compute the potential intersection points $(\nu_j, \mu_j) = \ell_0 \cap \ell_j$ which amounts to calculating $\nu_{-n}, \nu_{-n+1}, \ldots, \nu_{-1}, \nu_1, \nu_2, \cdots, \nu_n$ where

$$\nu_j = \frac{\mathcal{M} u_{|j|} + \mathcal{U}\mathrm{sign}(j)}{\mathcal{Q} u_{|j|} - \mathcal{U} q_{|j|}} \ . \tag{17}$$

The lines for which the denominator is zero correspond to infeasible intersection and can be discarded. The smallest value ν_j which is larger than the current traced value of ν corresponds to the next line intersecting ℓ_0.

While the above description is mathematically sound, we devised an equivalent intersection inspection scheme which is more numerically stable and efficient. We keep track of partition I_-, I_0, I_1 through the vector,

$$s_j = \begin{cases} -1 & j \in I_- \\ 0 & j \in I_0 \\ +1 & j \in I_+ \end{cases} \ .$$

Initially $s_1 = s_2 = \cdots = s_n = 0$. What kind of intersection does ℓ_0 have with ℓ_j? Recall that $\frac{\mathcal{Q}}{\mathcal{U}}$ is the slope of ℓ_0 while $\frac{q_{|j|}}{u_{|j|}}$ is the slope of ℓ_j. Thus $\frac{\mathcal{Q}}{\mathcal{U}} > \frac{q_{|j|}}{u_{|j|}}$ means that the $|j|$'th constraint is moving "up" from I_- to I_0 or from I_0 to I_+. When $\frac{\mathcal{Q}}{\mathcal{U}} < \frac{q_{|j|}}{u_{|j|}}$ the $|j|$'th constraint is moving "down" from I_+ to I_0 or from I_0 to I_-. See also Fig. 4 for an illustration of the possible transitions between the sets. For instance, the slope of $\mu(\nu)$ on the bottom left part of the figure is larger than the slope the line it intersects. Since this line defines the boundary between I_- and I_0, we transition from I_- to I_0. We need only consider $1 \le |j| \le n$ of the following types. Moving "up" from I_- to I_0 requires

$$s_{|j|} = -1 \qquad j < 0 \qquad \mathcal{Q} u_{|j|} - \mathcal{U} q_{|j|} > 0 \ .$$

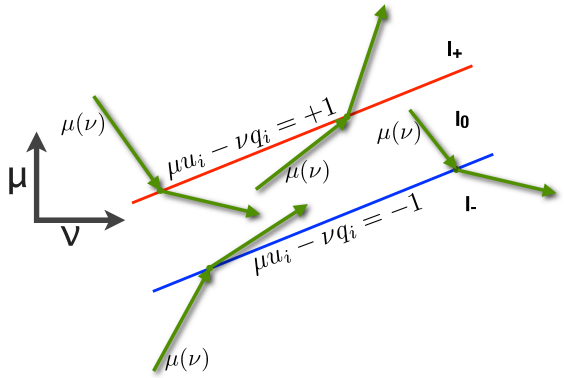


Figure 4: Illustration of the possible intersections between $\mu(\nu)$ and ℓ_j and the corresponding transition between the sets $I_\pm, I_0$.

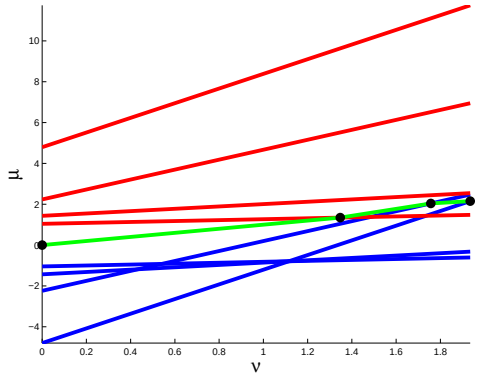


Figure 5: The result of the homotopy tracking for a 4 dimensional problem. The lines ℓ_j for $j < 0$ are drawn in blue and for $j > 0$ in red. The function $\mu(\nu)$ is drawn in green and its change points in black. Note that although the dimension is 4 the number of change points is rather small and does not exceed 4 either in this simple example.

Similarly, moving "down" from I_+ to I_0 requires

$$s_{|j|} = 1 \qquad j > 0 \qquad \mathcal{Q}u_{|j|} - \mathcal{U}q_{|j|} < 0 \ .$$

Finally, moving "up" or "down" from I_0 entails

$$s_{|j|} = 0 \qquad j(\mathcal{Q}u_{|j|} - \mathcal{U}q_{|j|}) > 0 \ .$$

If there are no eligible ν_j's, we have finished traversing $\mu()$. Otherwise let index j belong to the the smallest eligible ν_j. Infinite accuracy guarantees that $\nu_j \geq \nu$. In practice we perform the update

$$\begin{aligned}
\nu &\leftarrow \max(\nu, \nu_j) \\
\mathcal{M} &\leftarrow \mathcal{M} + \operatorname{sign}(\mathcal{Q}u_{|j|} - \mathcal{U}q_{|j|})\, m_{|j|} \\
\mathcal{U} &\leftarrow \mathcal{U} + \left(2\left|s_{|j|}\right| - 1\right)\, m_{|j|}\, u_{|j|} \\
\mathcal{Q} &\leftarrow \mathcal{Q} + \left(2\left|s_{|j|}\right| - 1\right)\, m_{|j|}\, q_{|j|} \\
s_j &\leftarrow s_j + \operatorname{sign}(\mathcal{Q}u_{|j|} - \mathcal{U}q_{|j|}) \ .
\end{aligned}$$

We are done with the tracking process when I_0 is empty, i.e. for all j $s_j \neq 0$.

The local homotopy algorithm takes $O(n)$ memory and $O(nk)$ operations where k is the number of change points in the function $\mu(\nu)$. This algorithm is simple to implement, and when k is relatively small it is efficient. An illustration of the tracking result, $\mu(\nu)$, along with the lines $\ell_{\pm j}$, that provide a geometrical description of the problem, is given in Fig. 5.

7 Uniform Prior

We chose to denote the prior distribution as $\boldsymbol{u}$ to underscore the fact that in the case of no prior knowledge $\boldsymbol{u}$ is the *uniform* distribution,

$$\mathrm{u} \stackrel{\text{def}}{=} u_j = \left(\sum_{i=1}^{n} m_i\right)^{-1} \ .$$

In this case the objective function amounts to the negative entropy and by flipping the sign of the objective we obtain the classical maximum entropy problem. The fact that the prior probability is the same for all possible observations infuses the problem with further structure which we show how to exploit in this section. Needless to say though that all the results we obtained thus far are still valid.

Let us consider a point (ν, μ) on the boundary between I_0 and I_+, namely, there exist a line ℓ_{+i} such that,

$$\mu u_i - \nu q_i = \mu \mathrm{u} - \nu q_i = 1 \ .$$

By definition, for any $j \in I_0$ we have

$$\mu u_j - \nu q_j = \mu \mathrm{u} - \nu q_j < 1 = \mu \mathrm{u} - \nu q_i \ .$$

Thus, $q_i < q_j$ for all $j \in I_0$ which implies that

$$m_j \,\mathrm{u}\, q_j \ > \ m_j \,\mathrm{u}\, q_i \ . \tag{18}$$

Summing over $j \in I_0$ we get that

$$\mathcal{Q}\,\mathrm{u} \ = \ \sum_{j \in I_0} m_j \, q_j \,\mathrm{u} \ > \ \sum_{j \in I_0} m_j \,\mathrm{u}\, q_i \ = \ \mathcal{U} q_i \ ,$$

hence,

$$\frac{q_i}{u_i} = \frac{q_i}{\mathrm{u}} < \frac{\mathcal{Q}}{\mathcal{U}}$$

and we must be moving "up" from I_0 to I_+ when the line ℓ_0 hits ℓ_i. Similarly we must be moving "down" from when ℓ_0 hits on the boundary between I_0 and I_-. We summarize these properties in the following theorem.

Theorem 7.1 *When the prior distribution $\boldsymbol{u}$ is uniform, $I_-(\nu)$ and $I_+(\nu)$ are monotonically nondecreasing and $I_0(\nu)$ is monotonically nonincreasing in $\nu > 0$. Further, the piecewise linear function $\mu(\nu)$ consists of at most $n+1$ line segments.*

The homotopy tracking procedure when the prior is uniform is particularly simple and efficient. Intuitively, there is a sole condition which controls the order in which indices would enter $I_\pm$ from I_0, which is simply how "far" each q_i is from u, the single prior value. Therefore, the algorithm starts by sorting $\boldsymbol{q}$. Let $q_{\pi_1} > q_{\pi_2} > \cdots > q_{\pi_n}$ denote the sorted vector. Instead of maintaining the vector of set indicators $\boldsymbol{s}$, we merely maintain two indices j_- and j_+ which designate the size of I_- and I_+ that were constructed thus far. Due to the monotonicity property of the sets $I_\pm$ as ν grows, the two sets can be written as, $I_- = \{\pi_j \,|\, 1 \le j < j_-\}$ and $I_+ = \{\pi_j \,|\, j_+ < j \le n\}$. The homotopy tracking procedure starts as before with $\nu = 0$, $\mathcal{M} = 0$, $\mathcal{U} = \mathcal{Q} = 1$. We also set $j_- = 1$ and $j_+ = n$ which by definition imply that $I_\pm$ are empty and $I_0 = [n]$. In each tracking iteration we need to compare only two values which we compactly denote as,

$$\nu_\pm = \frac{\mathcal{M}\mathrm{u} \pm \mathcal{U}}{\mathcal{Q}\mathrm{u} - \mathcal{U} q_{\pi_{j_\pm}}} \ .$$

When $\nu_- \le \nu_+$ we just encountered a transition from I_0 to I_- and as we encroach I_- we perform the updates, $\nu \leftarrow \nu_-$, $\mathcal{M} \leftarrow \mathcal{M} - m_{\pi_{j_-}}$, $\mathcal{U} \leftarrow \mathcal{U} - m_{\pi_{j_-}}\mathrm{u}$, $\mathcal{Q} \leftarrow \mathcal{Q} - m_{\pi_{j_-}} q_{\pi_{j_-}}$, $j_- \leftarrow j_- + 1$. Similarly when $\nu_- > \nu_+$ we perform the updates $\nu \leftarrow \nu_+$, $\mathcal{M} \leftarrow \mathcal{M} + m_{\pi_{j_+}}$, $\mathcal{U} \leftarrow \mathcal{U} - m_{\pi_{j_+}}\mathrm{u}$, $\mathcal{Q} \leftarrow \mathcal{Q} - m_{\pi_{j_+}} q_{\pi_{j_+}}$, $j_+ \leftarrow j_+ - 1$.

The tracking process stops when $j_- > j_+$ as we exhausted the transitions out of the set I_0 which becomes empty. Homotopy tracking for a uniform prior takes $O(n)$ memory and $O(n \log(n))$ operations and is very simple to implement.

We also devised a global homotopy tracking algorithms that requires a priority queue which facilitates insertions, deletions, and finding the largest element in the queue in $O(\log(n))$ time. The algorithm requires $O(n)$ memory and $O(n^2 \log(n))$ operations. Clearly, if the number of line segments constituting $\mu(\nu)$ is greater than $n \log(n)$ (recall that the upper bound is $O(n^2)$) then the global homotopy procedure is faster than the local one. However, as we show in Sec. 8, in practice the number of line segments is merely linear and it thus suffices to use the local homotopy tracking algorithm.

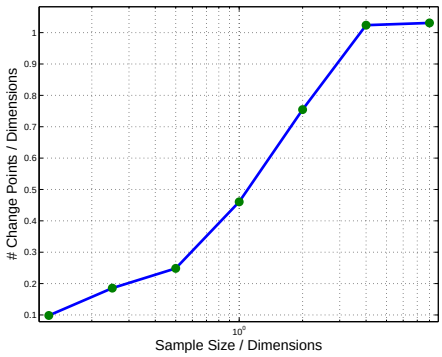


Figure 6: The number of line-segments in the homotopy as a function of the number of samples used to build the observed distribution $\boldsymbol{q}$.

8 Number of line segments in practice

The focus of the paper is the design and analysis of a novel homotopy method for maximum entropy problems. We thus left with relatively little space to discuss the empirical aspects of our approach. In this section we focus on one particular experimental facet that underscores the usability of our apparatus. We briefly discuss current natural language applications that we currently work on in the next section.

The practicality of our approach hinges on the number of line segments that occur in practice. Our bounds indicate that this number can scale quadratically with the dimension, which would render the homotopy algorithm impractical when the size of the alphabet is larger than a few thousands. We therefore extensively tested the *actual* number of line segments in the resulting homotopy when $\boldsymbol{u}$ and $\boldsymbol{q}$ are Zipf (1949) distributions. We used an alphabet of size $50,000$ in our experiments. The distribution $\boldsymbol{u}$ was set to be the Zipf distribution with an offset parameter of 2, that is, $u_i \sim 1/(i+2)$. We defined a "mother" distribution for $\boldsymbol{q}$, denoted $\bar{\boldsymbol{q}}$, which is

a plain Zipf distribution without an offset, namely $\bar{q}_i \sim 1/i$. We then sampled $n/2^l$ letters according to the distribution $\bar{\boldsymbol{q}}$ where $l \in -3, \ldots, 3$. Thus the smallest sample was $n/2^3 = 6,250$ and the largest sample was $n/3^{-3} = 40,000$. Based on the sample we defined the observed distribution $\boldsymbol{q}$ such that q_i is proportional to the number of times the i'th letter appeared in the sample. We repeated the process 100 times for each sample size and report average results. Note that when the sample is substantially smaller than the dimension the observed distribution $\boldsymbol{q}$ tends to be "simple" as it consists of many zero components. In Fig. 6 we depict the average number line segments for each sample size. When the sample size is one eighth of the dimension we average st most $0.1\,n$ line segments. More importantly, even when the size of the sample is fairly large, the number of lines segments is linear in the dimension with a constant close to one. We also performed experiments with large sample sizes for which the empirical distribution $\boldsymbol{q}$ is very close to the mother distribution $\bar{\boldsymbol{q}}$. We seldom found that the number of line segments exceeds $4n$ and the mode is around $2n$. These findings render our approach usable even in the very large natural language applications.

9 Conclusions

We presented a novel efficient apparatus for tracking the entire relaxation path of maximum entropy problems. We currently study natural language processing applications. In particular, we are in the process of devising homotopy methods for domain adaptation Blitzer (2008) and language modeling based on context tree weighting (Willems et al., 1995). We also examine generalization of our approach in which the relative entropy objective is replaced with a separable Bregman (Censor and Zenios, 1997) function. Such a generalization is likely to distill further connections to the other homotopy methods, in particular the least angle regression algorithm of Efron et al. (2004) and homotopy methods for the Lasso in general (Osborne et al., 2000). We also plan to study separable Bregman functions in order to derive entire path solutions for less explored objectives such as the Itakura-Saito spectral distance (Rabiner and Juang, 1993) and distances especially suited for natural language processing.

References

A.L. Berger, S.A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22 (1):39–71, 1996.

John Blitzer. *Domain Adaptation of Natural Language Processing Systems*. PhD thesis, University of Pennsylvania, 2008.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, New York, NY, USA, 1997.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179–190, 1997.

M. Dudík, S. J. Phillips, and R. E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, 8:1217–1260, June 2007.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

Edwin T. Jaynes. Prior probabilities. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4 (3):227–241, September 1968.

Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2): 319–337, 2000.

L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

George K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

A Non-negative Matrix Factorization Based Approach for Active Dual Supervision from Document and Word Labels

Chao Shen and **Tao Li**
School of Computing and Information Sciences
Florida International University
Miami, FL 33199 USA
{cshen001,taoli}@cs.fiu.edu

Abstract

In active dual supervision, not only informative examples but also features are selected for labeling to build a high quality classifier with low cost. However, how to measure the informativeness for both examples and feature on the same scale has not been well solved. In this paper, we propose a non-negative matrix factorization based approach to address this issue. We first extend the matrix factorization framework to explicitly model the corresponding relationships between feature classes and examples classes. Then by making use of the reconstruction error, we propose a unified scheme to determine which feature or example a classifier is most likely to benefit from having labeled. Empirical results demonstrate the effectiveness of our proposed methods.

1 Introduction

Active learning, as an effective paradigm to optimize the learning benefit from domain experts' feedback and to reduce the cost of acquiring labeled examples for supervised learning, has been intensively studied in recent years (McCallum and Nigam, 1998; Tong and Koller, 2002; Settles, 2009). Traditional approaches for active learning query the human experts to obtain the labels for intelligently chosen data samples. However, in text classification where the input data is generally represented as document-word matrices, human supervision can be obtained on both documents and words. For example, in sentiment analysis of product reviews, human labelers can label reviews as positive or negative, they can also label the words that elicit positive sentiment (such as "sensational" and "electrifying") as positive and words that evoke negative sentiment (such as "depressed" and "unfulfilling") as negative. Recent work has demonstrated that labeled words (or feature supervision) can greatly reduce the number of labeled samples for building high-quality classifiers (Druck et al., 2008; Zaidan and Eisner, 2008). In fact, different kinds of supervision generally have different acquisition costs, different degrees of utility and are not mutually redundant (Sindhwani et al., 2009). Ideally, effective active learning schemes should be able to utilize different forms of supervision.

To incorporate the supervision on words and documents at same time into the active learning scheme, recently an active dual supervision (or dual active learning) has been proposed (Melville and Sindhwani, 2009; Sindhwani et al., 2009). Comparing with traditional active learning which aims to select the most "informative" examples (e.g., documents) for domain experts to label, active dual supervision selects both the "informative" examples (e.g., documents) and features (e.g., words) for labeling. For active dual supervision to be effective, there are three important components: a) an underlying learning mechanism that is able to learn from both the labeled examples and features (i.e., incorporating supervision on both examples and features); b) methods for estimating the value of information for example and feature labels; and c) a scheme that should be able to trade-off the costs and benefits of the different forms of supervision since they have different labeling costs and different benefits.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 949–958,
Edinburgh, Scotland, UK, July 27–31, 2011.

In Sindhwani et al.'s initial work on active dual supervision (Sindhwani et al., 2009), a transductive bipartite graph regularization approach is used for learning from both labeled examples and features. In addition, uncertainty sampling and experimental design are used for selecting informative examples and features for labeling. To trade-off between different types of supervision, a simple probabilistic interleaving scheme where the active learner probabilistically queries the example oracle and the feature oracle is used. One problem in their method is that *the values of acquiring the feature labels and the example labels are not on the same scale*.

Recently, Li et al. (Li et al., 2009) proposed a dual supervision method based on constrained non-negative tri-factorization of the document-term matrix where the labeled features and examples are naturally incorporated as sets of constraints. Having a framework for incorporating dual-supervision based on matrix factorization, gives rise to the natural question of *how to perform active dual supervision in this setting*. Since rows and columns are treated equally in estimating the errors of matrix factorization, another question is can we address *the scaling issue in comparing the value of feature labels and example labels*.

In this paper, we study the problem of active dual supervision using non-negative matrix tri-factorization. Our work is based on the dual supervision framework using constrained non-negative tri-factorization proposed in (Li et al., 2009). We first extend the framework to explicitly model the corresponding relationships between feature classes and example classes. Then by making use of the reconstruction error criterion in matrix factorization, we propose a unified scheme to evaluate the value of feature and example labels. Instead of comparing the estimated performance increase of new feature labels or example labels, our proposed scheme assumes that a better supervision (a feature label or a example label) should lead to a more accurate reconstruction of the original data matrix. In our proposed scheme, *the value of feature labels and example labels is computed on the same scale*. The experiments show that our proposed unified scheme to query selection (i.e., feature/example selection for labeling) outperforms the interleaving schemes and the scheme based on expected log gain.

The rest of this paper is organized as follows: the related work is discussed in Section 2, and the dual supervision framework based on non-negative matrix tri-factorization is introduced in Section 3. We extend non-negative matrix tri-factorization to active learning settings in Section 4, and propose a unified scheme for query selection in Section 5. Experiments are presented in Section 6, and finally Section 7 concludes the paper.

2 Related Work

We point the reader to a recent report (Settles, 2009) for an in-depth survey on active learning. In this section, we briefly cover related work to position our contributions appropriately.

Active Learning/Active Dual Supervision Most prior work in active learning has focused on pooled-based techniques, where examples from an unlabeled pool are selected for labeling (Cohn et al., 1994). With the study of learning from labeled features, many research efforts on active learning with feature supervision are also reported (Melville et al., 2005; Raghavan et al., 2006). (Godbole et al., 2004) proposed the notion of feature uncertainty and incorporated the acquired feature labels into learning by creating one-term mini-documents. (Druck et al., 2009) performed active learning via feature labeling using several uncertainty reduction heuristics using the learning model developed in (Druck et al., 2008). (Sindhwani et al., 2009) studied the problem of active dual supervision from examples and features using a graph-based dual supervision method with a simple probabilistic method for interleaving feature labels and example labels. In our work, we develop our active dual supervision framework using constrained non-negative tri-factorization and also propose a unified scheme to evaluate the value of feature and example labels. We note the very recent work of (Attenberg et al., 2010), which proposes a unified approach for the dual active learning problem using expected utility where the utility is defined as the log gain of the classification model with a new labeled document or word. Conceptually, our proposed unified scheme is a special case of the expected utility framework where the utility is computed using the matrix reconstruction error. The utility based on the log gain of the classification

model may not be reliable as small model changes resulted from a single additional example label or feature label may not be reflected in the classification performance (Attenberg et al., 2010). The empirical comparisons show that our proposed unified scheme based on reconstruction error outperforms the expected log gain.

Dual Supervision Note that a learning method that is capable of performing dual supervision (i.e., learning from both labeled examples and features) is the basis for active dual supervision. Dual supervision is a relatively new area of research and few methods have been developed for dual supervision. In (Sindhwani and Melville, 2008; Sindhwani et al., 2008), a bipartite graph regularization model (GRADS) is used to diffuse label information along both sides of the document-term matrix and to perform dual supervision for semi-supervised sentiment analysis. Conceptually, their model implements a co-clustering assumption closely related to Singular Value Decomposition (see also (Dhillon, 2001; Zha et al., 2001) for more on this perspective). In (Sandler et al., 2008), standard regularization models are constrained using graphs of word co-occurrences. In (Melville et al., 2009), Naive Bayes classifier is extended, where the parameters, the conditional word distributions given the classes, are estimated by combining multiple sources, e.g. document labels and word labels. Our work is based on the dual supervision framework using constrained non-negative tri-factorization.

3 Learning with Dual Supervision via Tri-NMF

Our dual supervision model is based on non-negative matrix tri-factorization (Tri-NMF), where the non-negative input document-word matrix is approximated by 3 factor matrices as $X \approx GSF^T$, in which, X is an $n \times m$ document-term matrix, G is an $n \times k$ non-negative orthogonal matrix representing the probability of generating a document from a document cluster, F is an $m \times k$ non-negative orthogonal matrix representing the probability of generating a word from a word cluster, and S is a $k \times k$ non-negative matrix providing the relationship between document cluster space and word cluster space.

While Tri-NMF is first applied in co-clustering, Li et al. (Li et al., 2009) extended it to incorporate labeled words and documents as dual supervision via two loss terms in the objective function of Tri-NMF as following:

$$\begin{aligned} \min_{F,G,S} \quad & \|X - GSF^T\|^2 \\ & +\alpha \,\mathrm{trace}[(F - F_0)^T C_1 (F - F_0)] \\ & +\beta \,\mathrm{trace}[(G - G_0)^T C_2 (G - G_0)]. \end{aligned} \tag{1}$$

Here, $\alpha > 0$ is a parameter which determines the extent to which we enforce $F \approx F_0$ to its labeled rows. C_1 is a $m \times m$ diagonal matrix whose entry $(C_1)_{ii} = 1$ if the row of F_0 is labeled, that is, the class of the i-th word is known and $(C_1)_{ii} = 0$ otherwise. $\beta > 0$ is a parameter which determines the extent to which we enforce $G \approx G_0$ to its labeled rows. C_2 is a $n \times n$ diagonal matrix whose entry $(C_2)_{ii} = 1$ if the row of G_0 is labeled, that is, the category of the i-th document is known and $(C_2)_{ii} = 0$ otherwise. The squared loss terms ensure that the solution for G, F in the otherwise unsupervised learning problem be close to the prior knowledge G_0, F_0. So the partial labels on documents and words can be described using G_0 and F_0, respectively.

4 Dual Supervision with Explicit Class Alignment

4.1 Modeling the Relationships between Word Classes and Document Classes

In the solution to Equation 1, we have $S = G^T X F$, or

$$S_{lk} = g_l^T X f_k = \frac{1}{|R_l|^{1/2} |C_k|^{1/2}} \sum_{i \in R_l} \sum_{j \in C_k} X_{ij}, \tag{2}$$

where $|R_l|$ is the size of the l-th document class, and $|C_k|$ is the size of the k-th word class (Ding et al., 2006). Note that S_{lk} represents properly normalized within-class sum of weights ($l = k$) and between-class sum of weights ($l \neq k$). So, S represents the relationship between the classes over documents and the classes over words. Under the assumption that the i-th document class should correspond to the i-th word class, S should be an approximate diagonal matrix, since the documents of i-th class is more likely to contain the words of the i-th class. Note

that S is not an exact diagonal matrix, since a document of one class apparently can use words from other classes (especially G and F are required to be approximately orthogonal, which means the classification is rigorous). However, in Equation 1, there are no explicit constraints on the relationship between word classes and document classes. Instead, the relationship is established and enforced implicitly using existing labeled documents and words.

In active learning, the set of starting labeled documents or words is small, and this may generate an ill-formed S, leading to an incorrect alignment of word classes and document classes. To explicitly model the relationships between word classes and document classes, we constrain the shape of S via an extra loss term in the objective function as follows:

$$\begin{aligned}\min_{F,G,S} \quad & \|X - GSF^T\|^2 \\ & +\alpha \operatorname{trace}[(F - F_0)^T C_1 (F - F_0)] \\ & +\beta \operatorname{trace}[(G - G_0)^T C_2 (G - G_0)] \\ & +\gamma \operatorname{trace}[(S - S_0)^T (S - S_0)]\end{aligned} \tag{3}$$

where S_0 is a diagonal matrix.

How to Choose S_0 If there is no orthogonal constraint on F, G and I-divergence is used as the objective function, it can been shown that the factors of Tri-NMF have probabilistic interpretation (Ding et al., 2008; Shen et al., 2011):

$$\begin{aligned} F_{il} &= P(w = w_i | z_w = l), \\ G_{jk} &= P(d = d_j | z_d = k), \\ S_{kl} &= P(z_d = k, z_w = l), \end{aligned} \tag{4}$$

where w is word variable, d is document variable, and z_w, z_d are random variables indicating word class and document class respectively. F and G represent posterior distributions for words and documents, and S represents the joint distribution of document class and word class. With such an interpretation, S_0 can be easily decided in balanced classification problems with each diagonal entry equals to one over the number of classes.

However, in our setting of Tri-NMF, orthogonal constraints are enforced on F, G and Euclidean distance is used as the objective function. To precompute S_0, one way is to first solve the optimization problem Equation 1 with another constraint that S should be diagonal. Alternatively, to keep it simple, we ignore the known label information and just assume there exists a diagonal matrix S_0 and two orthogonal matrices G, F, that

$$GS_0F^T \approx X.$$

Then

$$\begin{aligned} \operatorname{trace}[XX^T] &\approx \operatorname{trace}[GS_0F^TFS_0^TG^T], \\ &= \operatorname{trace}[S_0S_0^TF^TFG^TG], \\ &= \operatorname{trace}[S_0S_0^T], \\ &= \textstyle\sum_k (S_0)_{kk}^2. \end{aligned} \tag{5}$$

So if a classification problem is balanced with K classes, S_0 can be estimated as following:

$$(S_0)_{kl} = \begin{cases} \sqrt{\frac{\operatorname{trace}[XX^T]}{K}} & l = k, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

4.2 Computing Algorithm

This optimization problem can be solved using the following update rules

$$\begin{aligned} G_{jk} &\leftarrow G_{jk} \frac{XFS + \beta C_2 G_0}{(GG^TXFS + \beta GG^TC_2G)_{jk}}, \\ S_{jk} &\leftarrow S_{jk} \frac{F^TX^TG + \gamma S_0}{(F^TFSG^TG + \gamma S)_{jk}}, \\ F_{jk} &\leftarrow F_{jk} \frac{X^TGS^T + \alpha C_1 F_0}{(FF^TX^TGS^T + \alpha C_1 F)_{jk}}. \end{aligned} \tag{7}$$

The algorithm consists of an iterative procedure using the above three rules until convergence.

Theorem 4.1 *The above iterative algorithm converges.*

Theorem 4.2 *At convergence, the solution satisfies the Karuch-Kuhn-Tucker (KKT) optimality condition, i.e., the algorithm converges correctly to a local optima.*

Theorem 4.1 can be proved using the standard auxiliary function approach (Lee and Seung, 2001).

Proof of Theorem 4.2: Proof for the update rules of G, F is the same as in (Li et al., 2009). Here we focus on the update rule of S. We want to minimize

$$\begin{aligned} L(S) = \quad & \|X - GSF^T\|^2 \\ & +\alpha \operatorname{trace}[(F - F_0)^T C_1 (F - F_0)] \\ & +\beta \operatorname{trace}[(G - G_0)^T C_2 (G - G_0)] \\ & +\gamma \operatorname{trace}[(S - S_0)^T (S - S_0)]. \end{aligned} \tag{8}$$

The gradient of L is

$$\frac{\partial L}{\partial S} = 2F^T FSG^T G - 2F^T X^T G + 2\gamma(S - S_0)$$

The KKT complementarity condition for the non-negativity of S_{jk} gives

$$[2F^T FSG^T G - 2F^T X^T G + 2\gamma(S - S_0)]_{jk} S_{jk} = 0.$$

This is the fixed point relation that local minima for S must satisfy, which is equivalent with the update rule of S in Equation 7. ∎

5 A Unified Scheme for Query Selection Using the Reconstruction Error

5.1 Introduction

An ideal active dual supervision scheme should be able to evaluate the value of acquiring labels for documents and words on the same scale. In the initial study of dual active supervision, different scores are used for documents and words (e.g. uncertainty for documents and certainty for words), and thus they are not on the same scale (Sindhwani et al., 2009). Recently, the framework of Expected Utility (Estimated Risk Minimization) is proposed in (Attenberg et al., 2010). At each step of the framework, the next word or document selected for labeling is the one that will result in the highest estimated improvement in classifier performance as defined as:

$$EU(q_j) = \sum_{k=1}^{K} P(q_j = c_k) U(q_j = c_k), \quad (9)$$

where K is the class number, $P(q_j = c_k)$ indicates the probability that q_j, j-th query (a word or document), belongs to the k-th class, and the $U(q_j = c_k)$ indicates the utility that q_j belongs to the k-th class. However, the choice of the utility measure is still a challenge.

5.2 Reconstruction Error

In our matrix factorization framework, rows and columns are treated equally in estimating the errors of matrix factorization, and the reconstruction error is thus a natural measure of utility. Let the current supervision knowledge be G_0, F_0. To select a new unlabeled document/word for labeling, we assume that a good supervision should lead to a good constrained factorization for the document-term matrix, $X \approx GSF^T$. If the new query q_j is a word and its label is k, then the new factorization is

$$\begin{aligned} & G^*_{j=k}, S^*_{j=k}, F^*_{j=k} \\ = \; & \arg\min_{G,S,F} \|X - GSF^T\|^2 \\ & + \alpha \,\mathrm{trace}[(G - G_0)^T C_2 (G - G_0)] \\ & + \beta \,\mathrm{trace}[(F - F_{0,j=k})^T C_1 (F - F_{0,j=k})] \\ & + \gamma \,\mathrm{trace}[(S - S_0)^T (S - S_0)], \end{aligned} \quad (10)$$

where $F_{0,j=k}$ is same as F_0 except that $F_{0,j=k}(j,k) = 1$. In other words, we obtained a new factorization using the labeled words. Similarly, if the new query q_j is a document, then the new factorization is

$$\begin{aligned} & G^*_{j=k}, S^*_{j=k}, F^*_{j=k} \\ = \; & \arg\min_{G,S,F} \|X - GSF^T\|^2 \\ & + \alpha \,\mathrm{trace}[(G - G_{0,j=k})^T C_2 (G - G_{0,j=k})] \\ & + \beta \,\mathrm{trace}[(F - F_0)^T C_1 (F - F_0)] \\ & + \gamma \,\mathrm{trace}[(S - S_0)^T (S - S_0)], \end{aligned} \quad (11)$$

where $G_{0,j=k}$ is same as G_0 except that $G_{0,j=k}(j,k) = 1$. In other words, we obtained a new factorization using the labeled documents. Then the new reconstruction error is

$$RE(q_j = k) = \|X - G^*_{j=k} S^*_{j=k} F^*_{j=k}\|^2. \quad (12)$$

So the expected utility of a document or word label query, q_j, can be computed as

$$EU(q_j) = \sum_{k=1}^{K} P(q_j = k) * (-RE(q_j = k)). \quad (13)$$

To calculate the $P(q_j = k)$, which is the posterior distribution for words or documents, probabilistic interpretation of Tri-NMF is abused. When a query q_j is a word, $P(q_j = k)$ is

$$\begin{aligned} & P(z_w = k | w = w_i) \\ \propto \; & P(w = w_i | z_w = k) \textstyle\sum_{j=1}^{K} P(z_w = k, z_d = j) \\ = \; & F_{ik} * \textstyle\sum_{j=1}^{K} S_{kj}, \end{aligned} \quad (14)$$

otherwise,

$$\begin{aligned} & P(z_d = k | d = d_i) \\ \propto \; & P(d = d_i | z_d = k) \textstyle\sum_{j=1}^{K} P(z_w = j, z_d = k) \\ = \; & G_{ik} * \textstyle\sum_{j=1}^{K} S_{jk}. \end{aligned} \quad (15)$$

5.3 Algorithm Description

Computational Improvement: It can be computationally intensive if the reconstruction error is computed for all unknown documents and words. Inspired by (Attenberg et al., 2010), we first select the top 100 unknown words that the current model is most certain about, and the top 100 unknown documents that the current model is most uncertain about. Then we identify the words or documents in this pool with the highest expected utility (reconstruction error). Equations 14 and 15 are used to perform the initial selection of top 100 unknown words and top 100 unknown documents.

Algorithm 1 Active Dual Supervision Algorithm Based on Matrix Factorization

INPUT: X, document-word matrix; F_0, current labeled words; G_0, current labeled documents; O, the oracle

OUTPUT: G, classification result for all documents in X

1. Get base factorization of X: G, S, F.
2. Active dual supervision

repeat

D is the set of top 100 unlabeled documents with most uncertainty;

W is the set of top 100 unlabeled words with most certainty;

$Q = D \cup W$;

for all $q \in Q$ **do**

for $k = 1$ to K **do**

Get $G^*_{q=k}, F^*_{q=k}, S^*_{q=k}$ by Equation 10 or Equation 11 according to whether the query q is a document or a word;

Calculate $EU(q)$ by Equation 13;

$q* = \arg\max_q EU(q)$;

Acquire new label of q^*, l from O;

$G, F, S = G^*_{q^*=l}, F^*_{q^*=l}, S^*_{q^*=l}$;

until stop criterion is met.

The overall algorithm procedure is described in Algorithm 1. First we iteratively use the updating rules of Equation 7 to obtain the factorization G, F, S based on initial labeled documents and words. Then to select a new query, for each unlabeled document or word in the pool and for each possible class, we compute the reconstruction error with new supervision (using the current factorization results as initialization values). It is efficient to compute a new factorization due to the sparsity of the matrices. The document-term matrix is typically very sparse with $z \ll nm$ non-zero entries while k is typically also much smaller than document number n, and word number m. By using sparse matrix multiplications and avoiding dense intermediate matrices, updating F, S, G each takes $O(k^2(m+n)+kz)$ time per iteration which scales linearly with the dimensions and density of the data matrix (Li et al., 2009). Empirically, the number of iterations that is needed to compute the new factorization is usually very small (less than 10).

6 Experiments

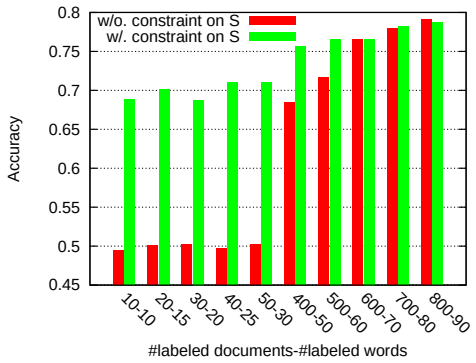
6.1 Experiments Settings

Three popular binary text classification datasets are used in the experiments: ibm-mac (1937 examples), baseball-hockey (1988 examples) and med-space (1972 examples) datasets. All of them are drawn from the 20-newsgroups text collection[1] where the task is to assign messages into the newsgroup in which they appeared. Top 1500 frequent words in each dataset are used as features in the binary vector representation. These datasets have labels for all the documents. For a document query, the oracle returns its label. We construct the word oracle in the same manner as in (Sindhwani et al., 2009): first compute the information gain of words with respect to the known true class labels in the training splits of a dataset, and then the top 100 words as ranked by information gain are assigned the label which is the class in which the word appears more frequently. To those words with labels, the word oracle returns its label; otherwise, the oracle returns a "don't know" response (no word label is obtained for learning, but the word is excluded from the following query selection).

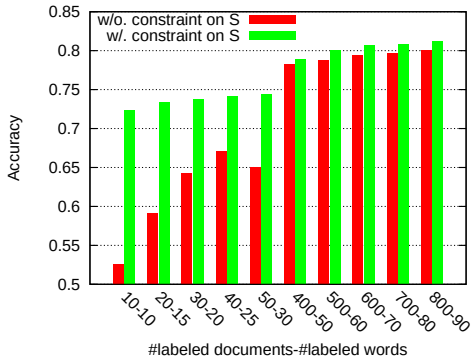
Results are averaged over 10 random training-test splits. For each split, 30% examples are used for testing. All methods are initialized by a random choice of 10 document labels and 10 word labels. For simplicity, we follow the widely used cost model (Raghavan and Allan, 2007; Druck et al.,

[1] http://www.ai.mit.edu/people/jrennie/20_newsgroups/

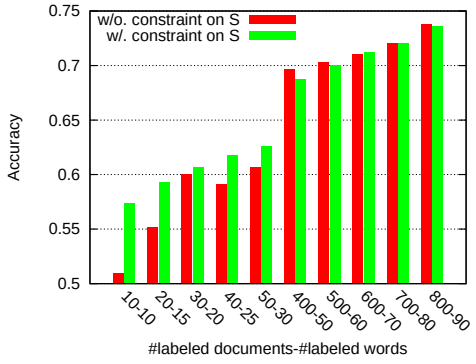
2008; Sindhwani et al., 2009) where features are roughly 5 times cheaper to label than examples, so we assume the cost is 1 for a word query and is 5 for a document query. We set $\alpha = \beta = 5, \gamma = 1$ for all the following experiments[2].



(a) baseball-hockey



(b) ibm-mac



(c) med-space

Figure 1: Comparing the performance of dual supervision via Tri-NMF w/ and w/o the constraint on S.

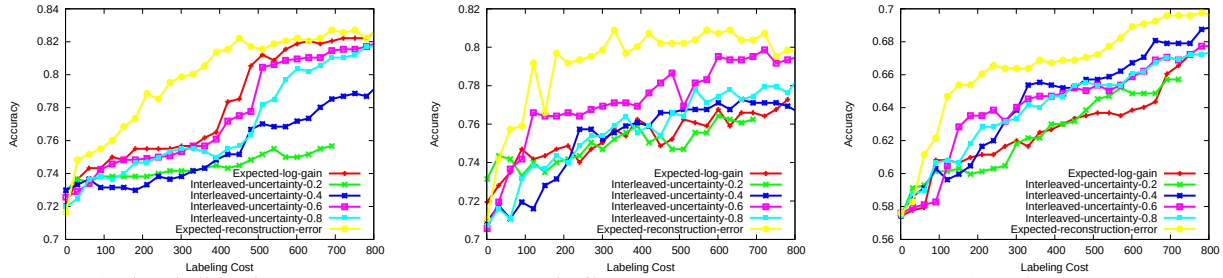
[2]We do not perform fine tuning on the parameters since the main objective of the paper is to demonstrate the effectiveness of matrix factorization based methods for dual active supervision. A vigorous investigation on the parameter choices is our further work.

6.2 Experimental Results

Effect of Constraints on S in Constrained Tri-NMF Figure 1 demonstrates the effectiveness of dual supervision with explicit class alignment via Tri-NMF as described in Section 4. When there are enough labeled documents and words, the constraints on S have a relative small impact on the performance of dual supervision. However, in the beginning phase of active learning, the labeled dataset can be small (such as 10 labeled documents and 10 labeled words). In this case, without the constraint of S, the matrix factorization may generate incorrect class alignment, thus lead to almost random classification results (around 50% accuracy), as shown in Figure 1, and further make unreasonable the following evaluation of queries.

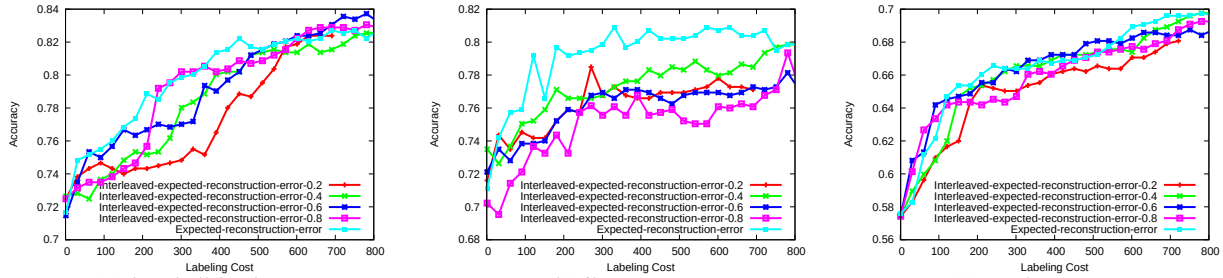
Comparing Query Selection Approaches Figure 2 compares our proposed unified scheme (denoted as *Expected-reconstruction-error*) with the following baselines using Tri-NMF as the classifier for dual supervision: (1). *Interleaved-uncertainty* which first selects feature query by certainty and sample query by uncertainty and then combines the two types of queries using an interleaving scheme. The interleaving probability (probability to select the query as a document) is set as 0.2, 0.4, 0.6 and 0.8. (2). *Expected-log-gain* which selects feature and sample query by maximizing the expected log gain. *Expected-reconstruction-error* outperforms interleaving schemes with all the different interleaving probability values with which we experimented. It also has a better performance than *Expected-log-gain*. Although log gain is a finer-grained utility measure of classifier performance than accuracy and has a good performance in the setting with a large set of starting labeled documents (e.g., 100 documents), it is not reliable especially in the setting with a small set of labeled data. Different from the *Expected-log-gain*, *Expected-reconstruction-error* estimates the utility using the matrix reconstruction error, making use of information of all documents and words, including those unlabeled.

Comparing Interleaving Scheme vs. the Unified Scheme To further demonstrate the benefit of the proposed unified scheme , we compare it with its interleaved version: *Interleaved-expected-*



(a) baseball-hockey (b) ibm-mac (c) med-space

Figure 2: Comparing the different query selection approaches in active learning via Tri-NMF with dual supervision.

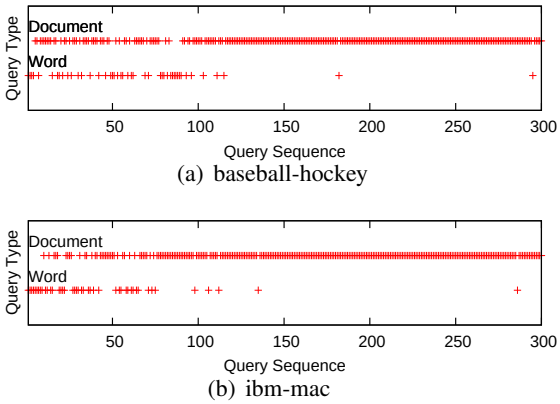


(a) baseball-hockey (b) ibm-mac (c) med-space

Figure 3: Comparing the unified and interleaving scheme based on reconstruction error.

construction-error which computes the utility of a query using the reconstruction error, but uses interleaving scheme to decide which type of query to select. We experiment with different interleaving probability values ranging from 0.2 to 0.8, which lead to quite different performance results. From Figure 3, the optimal interleaving probability value varies on different datasets. For example, the probability value of 0.8 is among the optimal interleaving probability values on baseball-hockey dataset but performs poorly on ibm-mac dataset. This observation also illustrates the need for a unified scheme, because of the difficulty in choosing the optimal interleaving probability value. Although the proposed unified scheme is not significantly better than its interleaving counterparts for all interleaving probability values on all datasets, it avoids the bad choices.

Figure 5 presents the sequence of different query types selected by our unified scheme and it clearly demonstrates the distribution patterns of different query types. At the beginning phase of active learning, word queries have much higher probabilities to be selected, which is consistent with the result of previous work: feature labels can be more effective than examples in text classification (Druck et al., 2008). And in the later learning phase, documents are more likely to be selected, since the number of words that can benefit the classification is much smaller than the effective documents.

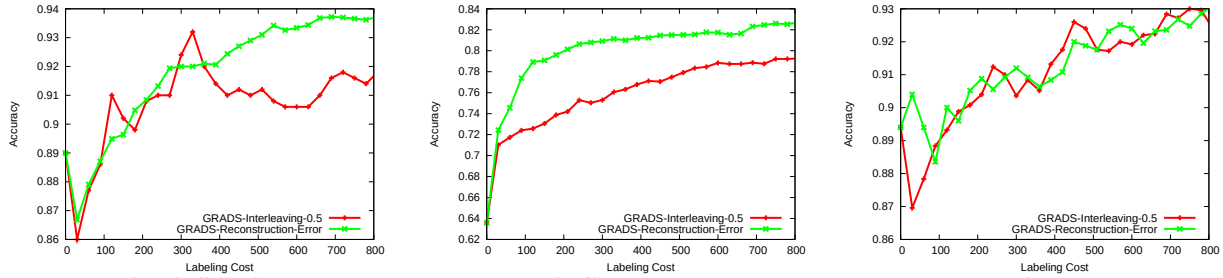


(a) baseball-hockey

(b) ibm-mac

Figure 5: Example of query sequence.

Reconstruction Error vs. Interleaving uncertainty using GRADS It should be pointed out that *our unified scheme for query selection based on reconstruction error does not rely on the estimation of model performance on training data and can be easily integrated with other dual supervision mod-*



(a) baseball-hockey (b) ibm-mac (c) med-space

Figure 4: GRADS with reconstruction error and interleaving uncertainty.

els such as GRADS (Sindhwani et al., 2008). Figure 4 shows the comparison of GRADS using the interleaved scheme with an interleaving probability of 0.5, and using our unified scheme based on reconstruction error. Among the 3 datasets we used, the reconstruction error based approach outperforms the interleaving scheme on baseball-hockey and ibm-mac, and has similar performance with the interleaving scheme on med-space.

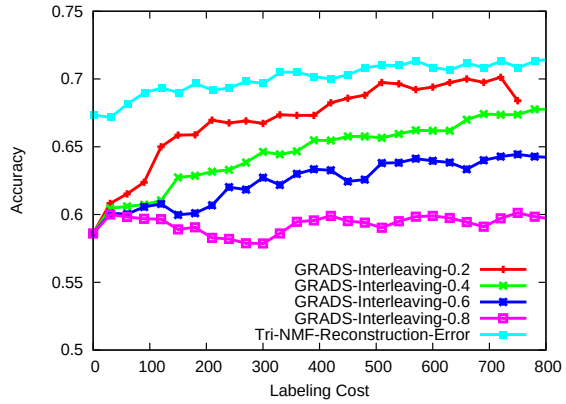


Figure 6: Comparing active dual supervision using matrix factorization with GRADS on sentiment analysis.

Comparing Active Dual Supervision Using Matrix Factorization with GRADS on Sentiment Analysis The sentiment analysis experiment is conducted on the movies review dataset (Pang et al., 2002), containing 1000 positive and 1000 negative movie reviews. The results are shown in Figure 6. The experimental results clearly demonstrate the effectiveness of our approach, denoted as *Tri-NMF-Reconstruction-Error*.

7 Conclusions

In this paper, we study the problem of active dual supervision, and propose a matrix tri-factorization based approach to address the issue, how to evaluate labeling benifit of different types of queries (examples or features) in the same scale. Following extending the nonnegative matrix tri-factorization to the active dual supervision setting, we use the reconstruction error to evaluate the value of feature and example labels. Experimental results show that our proposed approach outperforms existing methods.

Acknowledgement

The work is partially supported by NSF grants DMS-0915110, CCF-0830659, and HRD-0833093. We would like to thank Dr. Vikas Sindhwani for his insightful discussions and for sharing us with his GRADS code.

References

J. Attenberg, P. Melville, and F. Provost. 2010. A Unified Approach to Active Dual Supervision for Labeling Features and Examples. *Machine Learning and Knowledge Discovery in Databases*, pages 40–55.

D. Cohn, L. Atlas, and R. Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

I.S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM.

C. Ding, T. Li, W. Peng, and H. Park. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international*

conference on Knowledge discovery and data mining, pages 126–135. ACM.

C. Ding, T. Li, and W. Peng. 2008. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927.

G. Druck, G. Mann, and A. McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602. ACM.

G. Druck, B. Settles, and A. McCallum. 2009. Active learning by labeling features. In *Proceedings of the 2009 conference on Empirical methods in natural language processing*, pages 81–90. Association for Computational Linguistics.

S. Godbole, A. Harpale, S. Sarawagi, and S. Chakrabarti. 2004. Document classification through interactive supervision of document and term labels. *Knowledge Discovery in Databases: PKDD 2004*, pages 185–196.

D.D. Lee and H.S. Seung. 2001. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13.

T. Li, Y. Zhang, and V. Sindhwani. 2009. A non-negative matrix tri-factorization approach to sentiment classification with lexical prior knowledge. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL*, pages 244–252. Association for Computational Linguistics.

A.K. McCallum and K. Nigam. 1998. Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Citeseer.

P. Melville and V. Sindhwani. 2009. Active dual supervision: Reducing the cost of annotating examples and features. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 49–57. Association for Computational Linguistics.

P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. 2005. An expected utility approach to active feature-value acquisition. In *Proceedings of Fifth IEEE International Conference on Data Mining*. IEEE.

P. Melville, W. Gryc, and R.D. Lawrence. 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1275–1284. ACM.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the 2002 conference on Empirical methods in natural language processing*, pages 79–86. Association for Computational Linguistics.

H. Raghavan and J. Allan. 2007. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 79–86. ACM.

H. Raghavan, O. Madani, and R. Jones. 2006. Active learning with feedback on features and instances. *The Journal of Machine Learning Research*, 7:1655–1686.

T. Sandler, P.P. Talukdar, L.H. Ungar, and J. Blitzer. 2008. Regularized learning with networks of features. *Advances in Neural Information Processing Systems*, pages 1401–1408.

B. Settles. 2009. Active Learning Literature Survey. *Technical Report 1648*.

C. Shen, T. Li, and C. Ding. 2011. Integrating Clustering and Multi-Document Summarization by Bi-mixture Probabilistic Latent Semantic Analysis (PLSA) with Sentence Bases. In *Proceedings of the national conference on Artificial intelligence*. AAAI Press.

V. Sindhwani and P. Melville. 2008. Document-word co-regularization for semi-supervised sentiment analysis. In *Data Mining, Eighth IEEE International Conference on*, pages 1025–1030. IEEE.

V. Sindhwani, J. Hu, and A. Mojsilovic. 2008. Regularized co-clustering with dual supervision. *Advances in Neural Information Processing Systems*, 21.

V. Sindhwani, P. Melville, and R.D. Lawrence. 2009. Uncertainty sampling and transductive experimental design for active dual supervision. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 953–960. ACM.

S. Tong and D. Koller. 2002. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66.

Omar F. Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 conference on Empirical methods in natural language processing*, pages 31–40. Association for Computational Linguistics, October.

H. Zha, X. He, C. Ding, H. Simon, and M. Gu. 2001. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32. ACM.

Splitting Noun Compounds via Monolingual and Bilingual Paraphrasing: A Study on Japanese Katakana Words

Nobuhiro Kaji
Institute of Industrial Science
University of Tokyo, Tokyo, Japan
kaji@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa
Institute of Industrial Science
University of Tokyo, Tokyo, Japan
kitsure@tkl.iis.u-tokyo.ac.jp

Abstract

Word boundaries within noun compounds are not marked by white spaces in a number of languages, unlike in English, and it is beneficial for various NLP applications to split such noun compounds. In the case of Japanese, noun compounds made up of katakana words (i.e., transliterated foreign words) are particularly difficult to split, because katakana words are highly productive and are often out-of-vocabulary. To overcome this difficulty, we propose using monolingual and bilingual paraphrases of katakana noun compounds for identifying word boundaries. Experiments demonstrated that splitting accuracy is substantially improved by extracting such paraphrases from unlabeled textual data, the Web in our case, and then using that information for constructing splitting models.

1 Introduction

1.1 Japanese katakana words and noun compound splitting

Borrowing is a major type of word formation in Japanese, and numerous foreign words (proper names or neologisms etc.) are continuously being imported from other languages (Tsujimura, 2006). Most borrowed words in modern Japanese are transliterations[1] from English and they are referred to as *katakana words* because transliterated foreign words are primarily spelled by using katakana characters in the Japanese writing system.[2] Compounding is another type of word formation that is common in Japanese (Tsujimura, 2006). In particular, noun compounds are frequently produced by merging two or more nouns together. These two types of word formation yield a significant amount of katakana noun compounds, making Japanese a highly productive language.

In Japanese as well as some European and Asian languages (e.g., German, Dutch and Korean), constituent words of compounds are not separated by white spaces, unlike in English. In those languages, it is beneficial for various NLP applications to split such compounds. For example, compound splitting enables SMT systems to translate a compound on a word-by-word basis, even if the compound itself is not found in the translation table (Koehn and Knight, 2003; Dyer, 2009). In the context of IR, decompounding has an analogous effect to stemming, and it significantly improves retrieval results (Braschler and Ripplinger, 2004). In abbreviation recognition, the definition of an abbreviation is often in the form of a noun compound, and most abbreviation recognition algorithms assume that the definition is properly segmented; see e.g., (Schwartz and Hearst, 2003; Okazaki et al., 2008).

This has led NLP researchers to explore methods for splitting compounds, especially noun compounds, in various languages (Koehn and Knight, 2003; Nakazawa et al., 2005; Alfonseca et al., 2008a). While many methods have been presented, they basically require expensive linguistic resources to achieve high enough accuracy. For example, Alfonseca et al. (2008b) employed a word dictionary, which is obviously useful for this task. Other studies have suggested using bilingual resources such as parallel corpora (Brown, 2002; Koehn and Knight,

[1] Some researchers use the term transcription rather than transliteration (Breen, 2009). Our terminology is based on studies on machine transliteration (Knight and Graehl, 1998).

[2] The Japanese writing system has four character types: hiragana, katakana, kanji, and Latin alphabet.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 959–969,
Edinburgh, Scotland, UK, July 27–31, 2011.

2003; Nakazawa et al., 2005). The idea behind those methods is that compounds are basically split into constituent words when they are translated into English, where the compounded words are separated by white spaces, and hence splitting rules can be learned by discovering word alignments in bilingual resources.

The largest obstacle that makes compound splitting difficult is the existence of out-of-vocabulary words, which are not found in the abovementioned linguistic resources. In the Japanese case, it is known that katakana words constitute a large source of out-of-vocabulary words (Brill et al., 2001; Nakazawa et al., 2005; Breen, 2009). As we have discussed, katakana words are very productive, and thus we can no longer expect existent linguistic resources to have sufficient coverage. According to (Breen, 2009), as many as 20% of katakana words in news articles, which we think include less out-of-vocabulary words than Web and other noisy textual data, are out-of-vocabulary. Those katakana words often form noun compounds, and pose a substantial difficulty for Japanese text processing (Nakazawa et al., 2005).

1.2 Paraphrases as implicit word boundaries

To alleviate the errors caused by out-of-vocabulary words, we explored the use of unlabeled textual data for splitting katakana noun compounds. Since the amount of unlabeled text available is generally much larger than word dictionaries and other expensive linguistic resources, it is crucial to establish a methodology for taking full advantage of such easily available textual data. While several approaches have already been proposed, their accuracies are still unsatisfactory (section 2.1).

From a broad perspective, our approach can be seen as using paraphrases of noun compounds. As we will see in section 4 and 5, katakana noun compounds can be paraphrased into various forms that strongly indicate word boundaries within the original noun compound. This paper empirically demonstrates that splitting accuracy can be significantly improved by extracting such paraphrases from unlabeled text, the Web in our case, and then using that information for constructing splitting models.

Specifically, two types of paraphrases are investigated in this paper. Section 4 explores monolingual paraphrases that can be generated by inserting certain linguistic markers between constituent words of katakana noun compounds. Section 5, in turn, explores bilingual paraphrases (specifically, back-transliteration). Since katakana words are basically transliterations from English, back-transliterating katakana noun compounds is also useful for splitting. To avoid terminological confusion, monolingual paraphrases are simply referred to as paraphrases and bilingual paraphrases are referred to as back-transliterations hereafter.

We did experiments to empirically evaluate our method. The results demonstrated that both paraphrase and back-transliteration substantially improved the performance in terms of F_1-score, and the best performance was achieved when they were combined. We also confirmed that our method outperforms the previously proposed splitting methods by a wide margin. All these results strongly suggest the effectiveness of paraphrasing and back-transliteration for identifying word boundaries within katakana noun compounds.

2 Related Work

2.1 Compound splitting

A common approach to splitting compounds without expensive linguistic resources is an unsupervised method based on word or string frequencies estimated from unlabeled text (Koehn and Knight, 2003; Ando and Lee, 2003; Schiller, 2005; Nakazawa et al., 2005; Holz and Biemann, 2008). Amongst others, Nakazawa et al. (2005) also investigated ways of splitting katakana noun compounds. Although the frequency-based method generally achieves high recall, its precision is not satisfactory (Koehn and Knight, 2003; Nakazawa et al., 2005). Our experiments empirically compared our method with the frequency-based methods, and the results demonstrate the advantage of our method.

Our approach can be seen as augmenting discriminative models of compound splitting with large external linguistic resources, i.e., textual data on the Web. In a similar spirit, Alfonseca et al. (2008b) proposed the use of query logs for compound splitting.[3] Their experimental results, however, did not clearly

[3] Although they also proposed using anchor text, this slightly degraded the performance.

demonstrate their method's effectiveness. Without the query logs, the accuracy is reported to drop only slightly from 90.55% to 90.45%. In contrast, our experimental results showed statistically significant improvements as a result of using additional resources. Moreover, we used only textual data, which is easily available, unlike query logs.

Holz and Biemann (2008) proposed a method for splitting and paraphrasing German compounds. While their work is related to ours, their algorithm is a pipeline model and paraphrasing result is not employed during splitting.

2.2 Other research topics

Our study is closely related to word segmentation, which is an important research topic in Asian languages including Japanese. Although we can use existing word segmentation systems for splitting katakana noun compounds, it is difficult to reach the desired accuracy, as we will empirically demonstrate in section 6. One reason for this is that katakana noun compounds often include out-of-vocabulary words, which are difficult for the existing segmentation systems to deal with. See (Nakazawa et al., 2005) for a discussion of this point. From a word segmentation perspective, our task can be seen as a case study focusing on a certain linguistic phenomenon of particular difficulty. More importantly, we are unaware of any attempts to use paraphrases or transliterations for word segmentation in the same way as we do.

Recent studies have explored using paraphrase statistics for parsing (Nakov and Hearst, 2005a; Nakov and Hearst, 2005b; Bansal and Klein, 2011). Although these studies successfully demonstrated the usefulness of paraphrases for improving parsers, the connection between paraphrases and word segmentation (or noun compound splitting) was not at all discussed.

Our method of using back-transliterations for splitting katakana noun compounds (section 5) is closely related to methods for mining transliteration from the Web text (Brill et al., 2001; Cao et al., 2007; Oh and Isahara, 2008; Wu et al., 2009). What most differentiates these studies from our work is that their primary goal is to build a machine transliteration system or to build a bilingual dictionary itself; none of them explored splitting compounds.

Table 1: Basic features.

ID	Feature	Description
1	y_i	constituent word 1-gram
2	$y_{i-1}y_i$	constituent word 2-gram
3	$\textsc{Len}(y_i)$	#characters of y_i (1, 2, 3, 4, or $\geq$5)
4	$\textsc{Dict}(y_i)$	true if y_i is in the dictionary

3 Supervised Approach

The task we examine in this paper is splitting a katakana noun compound x into its constituent words, $\boldsymbol{y} = (y_1, y_2 \ldots y_{|\boldsymbol{y}|})$. Note that the output can be a single word, i.e., $|\boldsymbol{y}| = 1$. Since it is possible that the input is an out-of-vocabulary word, it is not at all trivial to identify a single word as such. A naive method would erroneously split an out-of-vocabulary word into multiple constituent words.

We formalize our task as a structure prediction problem that, given a katakana noun compound x, predicts the most probable splitting $\boldsymbol{y}^*$.

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(x)} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{y}),$$

where $\mathcal{Y}(x)$ represents the set of all splitting options of x, $\boldsymbol{\phi}(\boldsymbol{y})$ is a feature vector representation of $\boldsymbol{y}$, and $\boldsymbol{w}$ is a weight vector to be estimated from labeled data.

Table 1 summarizes our basic feature set. Features 1 and 2 are word 1-gram and 2-gram features, respectively. Feature 3 represents the length of the constituent word. $\textsc{Len}(y)$ returns the number of characters of y (1, 2, 3, 4, or $\geq$5). Feature 4 indicates whether the constituent word is registered in an external dictionary (see section 6.1). $\textsc{Dict}(y)$ returns true if the word y is in the dictionary.

In addition to those basic features, we also employ paraphrases and back-transliterations of katakana noun compounds as features. The features are detailed in sections 4 and 5, respectively.

We can optimize the weight vector $\boldsymbol{w}$ using an arbitrary training algorithm. Here we adopt the averaged perceptron algorithm for the sake of time efficiency (Freund and Schapire, 1999). The perceptron offers efficient online training, and it performs comparatively well with batch algorithms such as SVMs. Since we use only factored features (see table 1, section 4 and section 5), dynamic programming can be used to locate $\boldsymbol{y}^*$.

Table 2: Paraphrase rules and examples. The first column represents the type of linguistic marker to be inserted, the second column shows the paraphrase rules, and the last column gives examples.

Type	Rule	Example
Centered dot	$X_1X_2 \rightarrow X_1 \cdot X_2$	アンチョビパスタ → アンチョビ・パスタ (anchovy pasta) (anchovy · pasta)
Possessive marker	$X_1X_2 \rightarrow X_1$ の X_2	アンチョビパスタ → アンチョビの パスタ (anchovy pasta) (with anchovy) (pasta)
Verbal suffix	$X_1X_2 \rightarrow X_1$ する X_2 $X_1X_2 \rightarrow X_1$ した X_2	ダウンロードファイル→ダウンロードしたファイル (download file) (downloaded) (file)
Adjectival suffix	$X_1X_2 \rightarrow X_1$ な X_2 $X_1X_2 \rightarrow X_1$ 的 X_2 $X_1X_2 \rightarrow X_1$ 的な X_2	サプライズギフト → サプライズなギフト (surprise gift) (surprising) (gift)

4 Paraphrasing

In this section, we argue that paraphrases of katakana noun compounds provides useful information on word boundaries. Consequently, we propose using paraphrase frequencies as features for training the discriminative model.

4.1 Paraphrasing noun compounds

A katakana noun compound can be paraphrased into various forms, some of which provide information on the word boundaries within the original compound.

(1) a. アンチョビパスタ
(anchovy pasta)
b. アンチョビ・パスタ
(anchovy · pasta)
c. アンチョビの パスタ
(with anchovy) (pasta)

These examples are paraphrases of each other. (1a) is in the form of a noun compound, within which the word boundary is ambiguous. In (1b), on the other hand, a centered dot ・ is inserted between the constituent words. In the Japanese writing system, the centered dot is sometimes, but not always, used to separate long katakana compounds for the sake of readability. (1c) is the noun phrase generated from (1a) by inserting the possessive marker 'の', which can be translated as *with* in this context, between the constituent words. If we observe paraphrases of (1a) such as (1b) and (1c), we can guess that a word boundary exists between 'アンチョビ (anchovy)' and 'パスタ (pasta)'.

4.2 Paraphrase rules

The above discussion led us to use paraphrase frequencies estimated from Web text for splitting katakana noun compounds. For this purpose, we established the seven paraphrase rules illustrated in Table 2. The rules are in the form of $X_1X_2 \rightarrow X_1MX_2$, where X_1 and X_2 represent nouns, and M is a certain linguistic marker (e.g., the possessive marker 'の'). The left-hand term corresponds to a compound to be paraphrased and the right-hand term represents its paraphrase. For instance, X_1 = 'アンチョビ (anchovy)', X_2 = 'パスタ (pasta)', and M = 'の'. The paraphrase rules we use are based on the rules proposed by Kageura et al. (2004) for expanding complex terms, primarily noun compounds, into their variants.

4.3 Web-based frequency as features

We introduce a new feature using the paraphrase rules and Web text. As preprocessing, we use regular expressions to count the frequencies of all potential paraphrases of katakana noun compounds on the Web in advance.

(katakana)+ ・ (katakana)+
(katakana)+ の (katakana)+
(katakana)+ する (katakana)+
...

where (katakana) corresponds to one katakana character. Given a candidate segmentation $\boldsymbol{y}$ at test time, we generate paraphrases of the noun compound by setting $X_1 = y_{i-1}$ and $X_2 = y_i$, and applying the paraphrase rules. We then use $\log(F + 1)$, where F is the sum of the Web-based frequencies of the gen-

erated paraphrases, as the feature of the boundary between y_{i-1} and y_i.

As the feature value, we use the logarithmic frequency, rather than the raw frequency, for scaling. Since the other features have binary value, we found, in initial experiments, that the importance of this feature is overemphasized if we use the raw frequency. Note that we use $\log(F+1)$ rather than $\log F$ so as to avoid the feature value being zero when $F = 1$.

5 Back-transliteration

Most katakana words are transliterations from English, where words are separated by white spaces. It is, therefore, reasonable to think that back-transliterating katakana noun compounds into English would provide information on word boundaries, in a similar way to paraphrasing.

This section presents a method for extracting back-transliterations of katakana words from monolingual Web text, and establishing word alignments between those katakana and English words (Table 3). In what follows, the pair of katakana words and its English back-transliteration is referred to as a *transliteration pair*. If the transliteration pair is annotated with word alignment information as in Table 3, it is referred to as a *word-aligned transliteration pair*.

Using word-aligned transliteration pairs extracted from the Web text, we derive a binary feature indicating whether katakana word y_i corresponds to a single English word. Additionally, we derive another feature indicating whether a katakana word 2-gram $y_{i-1}y_i$ corresponds to an English word 2-gram.

5.1 Parenthetical expressions

In Japanese and other Asian languages, transliterated words are sometimes followed by their English back-transliterations inside parentheses:

(2) a. アメリカでは ‖ ジャンク (junk) フード (food) (junk food) と...

b. トラックバック ‖ スパム (spam) (spam) を撃退する...

where the underline indicates the Japanese text that is followed by English back-transliteration. We extract word-aligned transliteration pairs from such parenthetical expressions by establishing the correspondences between pre-parenthesis and in-parenthesis words.

Table 3: Word-aligned transliteration pairs. The number indicates the word alignment.

Japanese	**English**
ジャンク$_1$フード$_2$	junk$_1$ food$_2$
スパム$_3$	spam$_3$

To accomplish this, we have to resolve three problems: (a) English words inside parenthesis do not always provide a back-transliteration of the pre-parenthesis text, (b) the left boundary of the pre-parenthesis text, denoted as '‖' in the example, has to be identified, and (c) pre-parenthesis text, which is a katakana noun compound in our case, has to be segmented into words.

Although several studies have explored mining transliterations from such parenthetical expressions (Cao et al., 2007; Wu et al., 2009), the last problem has not been given much attention. In the past studies, the pre-parenthesis text is assumed to be correctly segmented by, typically, using existent word segmentation systems. This is, however, not appropriate for our purpose, because pre-parenthesis text is a katakana noun compound, which is hard for existing systems to handle, and hence the alignment quality is inevitably affected by segmentation errors.

To handle these three problems, we use the phonetic properties of the transliterations. For the purpose of explanation, we shall first focus on problem (c). Since transliterated katakana words preserve the pronunciation of the original English words to some extent (Knight and Graehl, 1998), we can discover the correspondences between substrings of the two languages based on phonetic similarity:

(3) a. [ジャン]$_1$[ク]$_2$[フー]$_3$[ド]$_4$

b. [jun]$_1$[k]$_2$ [foo]$_3$[d]$_4$

Note that these are the pre-parenthesis and in-parenthesis text in (2a). The substrings surrounded by square brackets with the same number correspond to each other. Given such a correspondence, we can segment the pre-parenthesis text (3a) according to its English counterpart (3b), in which words

Table 4: Example of the substring alignment $\mathcal{A}$ between f ='ジャンクフード' and e ='junkfood' ($|\mathcal{A}| = 4$).

(f_i, e_i)	$\log p(f_i, e_i)$
(ジャン, jun)	-10.767
(ク, k)	-5.319
(フー, foo)	-11.755
(ド, d)	-5.178

are separated by white space. We can recognize that the katakana string 'ジャンク', which is the concatenation of the first two substrings in (3a), forms a single word because it corresponds to the English word *junk*, and so on. Consequently, (3a) can be segmented into two words, 'ジャンク (junk)' and 'フード (food)'. The word alignment is trivially established.

For problems (a) and (b), we can also use the phonetic similarity between pre-parenthesis and in-parenthesis text. If the parenthetical expression does not provide the transliteration, or if the left boundary is erroneously identified, we can expect the phonetic similarity to become small. Such situations thus can be identified.

The remainder of this section details this approach. Section 5.2 presents a probabilistic model for discovering substring alignment such as (3). Section 5.3 shows how to extract word-aligned transliteration pairs by using the probabilistic model.

5.2 Phonetic similarity model

To establish the substring alignment between katakana and Latin alphabet strings, we use the probabilistic model proposed by (Jiampojamarn et al., 2007). Let f and e be katakana and alphabet strings, and $\mathcal{A}$ be the substring alignment between them. More precisely, $\mathcal{A}$ is a set of corresponding substring pairs (f_i, e_i) such that $f = f_1 f_2 \dots f_{|\mathcal{A}|}$ and $e = e_1 e_2 \dots e_{|\mathcal{A}|}$. The probability of such alignment is defined as

$$\log p(f, e, \mathcal{A}) = \sum_{(f_i, e_i) \in \mathcal{A}} \log p(f_i, e_i).$$

Since $\mathcal{A}$ is usually unobservable, it is treated as a hidden variable. Table 4 illustrates an example of the substring alignment between f ='ジャンクフード' and e ='junkfood', and the likelihood of each substring pair estimated in our experiment.

The model parameters are estimated from a set of transliteration pairs (f, e) using the EM algorithm. In the E-step, we estimate $p(\mathcal{A}|f, e)$ based on the current parameters. In the parameter estimation, we restrict both f_i and e_i to be at most three characters long. Doing this not only makes the E-step computationally efficient but avoids over-fitting by forbidding too-long substrings to be aligned. In the M-step, the parameter is re-estimated using the result of the E-step. We can accomplish this by using an extension of the forward-backward algorithm. See (Jiampojamarn et al., 2007) for details.

Given a new transliteration pair (f, e), we can determine the substring alignment as

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A}} \log p(f, e, \mathcal{A}).$$

In finding the substring alignment, a white space on the English side is used as a constraint, so that the English substring e_i does not span a white space.

5.3 Extracting word-aligned transliteration pairs

The word-aligned transliteration pairs are extracted using the phonetic similarity model, as follows.

First, candidate transliteration pairs (f, e) are extracted from the parenthetical expressions. This is done by extracting English words inside parentheses and pre-parenthesis text written in katakana. English words are normalized by lower-casing capital letters.

Second, we determine the left boundary by using the confidence score: $\frac{1}{N} \log p(f, e, \mathcal{A}^*)$, where N is the number of English words. The term $\frac{1}{N}$ prevents the score from being unreasonably small when there are many words. We truncate f by removing the leftmost characters one by one, until the confidence score exceeds a predefined threshold θ. If f becomes empty, the pair is regarded as a non-transliteration and discarded.

Finally, for the remaining pairs, the Japanese side is segmented and the word alignment is established according to $\mathcal{A}^*$. This results in a list of word-aligned transliteration pairs (Table 3).

6 Experiments and Discussion

We conducted experiments to investigate how the use of the paraphrasing and the back-transliteration

improves the performance of the discriminative model.

6.1 Experimental setting

To train the phonetic similarity model, we used a set of transliteration pairs extracted from the Wikipedia.[4] Since person names are almost always transliterated when they are imported from English into Japanese, we made use of the Wikipedia articles that belong to the *Living people* category. From the titles of those articles, we automatically extracted person names written in katakana, together with their English counterparts obtainable via the multilingual links provided by the Wikipedia. This yielded 17,509 transliteration pairs for training. In performing the EM algorithm, we tried ten different initial parameters and selected the model that achieved the highest likelihood.

The data for training and testing the perceptron was built using a Japanese-English dictionary EDICT.[5] We randomly extracted 5286 entries written in katakana from EDICT and manually annotated word boundaries by establishing word correspondences to their English transliterations. Since English transliterations are already provided by EDICT, the annotation can be trivially done by native speakers of Japanese. Using this data set, we performed 2-fold cross-validation for testing the perceptron. The number of iterations was set to 20 in all the experiments.

To compute the dictionary-based feature DICT(y) in our basic feature set, we used NAIST-jdic.[6] It is the largest dictionary used for Japanese word segmentation, and it includes 19,885 katakana words.

As Web corpora, we used 1.7 G sentences of blog articles. From the corpora, we extracted 14,966,205 (potential) paraphrases of katakana noun compounds together with their frequencies. We also extracted 151,195 word-aligned transliteration pairs. In doing this, we ranged the threshold θ in $\{-10, -20, \cdots -150\}$ and chose the value that performed the best ($\theta = -80$).

The results were evaluated using precision, recall, F_1-score, and accuracy. Precision is the number of correctly identified words divided by the number of all identified words, recall is the number of correctly identified words divided by the number of all oracle words, the F_1-score is their harmonic mean, and accuracy is the number of correctly split katakana noun compounds divided by the number of all the katakana noun compounds.

[4] http://ja.wikipedia.org/

[5] http://www.csse.monash.edu.au/~jwb/edict_doc.html

[6] http://sourceforge.jp/projects/naist-jdic

6.2 Baseline systems

We compared our system with three frequency-based baseline system, two supervised baselines, and two state-of-the-art word segmentation baselines. The first frequency-based baseline, UNIGRAM, performs compound splitting based on a word 1-gram language model (Schiller, 2005; Alfonseca et al., 2008b):

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(x)} \prod_i p(y_i),$$

where $p(y_i)$ represents the probability of y_i. The second frequency-based baseline, GMF, outputs the splitting option with the highest geometric mean frequency of the constituent words (Koehn and Knight, 2003):

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(x)} \mathrm{GMF}(\boldsymbol{y}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(x)} \Big\{ \prod_i f(y_i) \Big\}^{1/|\boldsymbol{y}|},$$

where $f(y_i)$ represents the frequency of y_i. The third frequency-based baseline, GMF2, is a modification of GMF proposed by Nakazawa et al. (2005). It is based on the following score instead of GMF($\boldsymbol{y}$):

$$\mathrm{GMF2}(\boldsymbol{y}) = \begin{cases} \mathrm{GMF}(\boldsymbol{y}) & (|\boldsymbol{y}| = 1) \\ \frac{\mathrm{GMF}(\boldsymbol{y})}{\frac{C}{N^l} + \alpha} & (|\boldsymbol{y}| \geq 2), \end{cases}$$

where C, N, and α are hyperparameters and l is the average length of the constituent words. Following (Nakazawa et al., 2005), the hyperparameters were set as C = 2500, N = 4, and α = 0.7. We estimated $p(y)$ and $f(y)$ from the Web corpora.

The first supervised baseline, AP, is the averaged perceptron model trained using only the basic feature set. The second supervised baseline, AP+GMF2 is a combination of AP and GMF2, which performed the best amongst the frequency-based baselines. Following (Alfonseca et al.,

Table 5: Comparison with baseline systems.

Type	System	P	R	F_1	Acc
Frequency	UNIGRAM	64.2	49.7	56.0	63.0
	GMF	42.9	62.0	50.7	47.5
	GMF2	67.4	76.0	71.5	72.5
Supervised	AP	81.9	82.5	82.2	83.4
	AP+GMF2	83.0	83.9	83.4	84.2
	PROPOSED	**86.4**	**87.4**	**87.1**	**87.6**
Word seg.	JUMAN	71.4	60.1	65.3	69.8
	MECAB	72.4	73.7	67.8	71.6

2008b), GMF2 is integrated into AP as two binary features indicating whether GMF2($\boldsymbol{y}$) is larger than any other candidates, and whether GMF2($\boldsymbol{y}$) is larger than the non-split candidate. Although Alfonseca et al. (2008b) also proposed using (the log of) the geometric mean frequency as a feature, doing so degraded performance in our experiment.

Regarding the two state-of-the-art word segmentation systems, one is JUMAN,[7] a rule-based word segmentation system (Kurohashi and Nagao, 1994), and the other is MECAB,[8] a supervised word segmentation system based on CRFs (Kudo et al., 2004). These two baselines were chosen in order to show how well existing word segmentation systems perform this task. Although the literature states that it is hard for existing systems to deal with katakana noun compounds (Nakazawa et al., 2005), no empirical data on this issue has been presented until now.

6.3 Splitting result

Table 5 compares the performance of our system (PROPOSED) with the baseline systems. First of all, we can see that PROPOSED clearly improved the performance of AP, demonstrating the effectiveness of using paraphrases and back-transliterations.

Our system also outperformed all the frequency-based baselines (UNIGRAM, GMF, and GMF2). This is not surprising, since the simple supervised baseline, AP, already outperformed the unsupervised frequency-based ones. Indeed similar experimental results were also reported by Alfonseca (2008a). An interesting observation here is the comparison between PROPOSED and AP+GMF2. It reveals that our approach improved the performance of AP more than the frequency-based method did. These results indicate that paraphrasing and back-transliteration are more informative clues than the simple frequency of constituent words. We would like to note that the higher accuracy of PROPOSED in comparison with the baselines is statistically significant ($p < 0.01$, McNemar's test).

The performance of the two word segmentation baselines (JUMAN and MECAB) is significantly worse in our task than in the standard word segmentation task, where nearly 99% precision and recall are reported (Kudo et al., 2004). This demonstrates that splitting a katakana noun compound is not at all a trivial task to resolve, even for the state-of-the-art word segmentation systems. On the other hand, PROPOSED outperformed both JUMAN and MECAB in this task, meaning that our technique can successfully complement the weaknesses of the existing word segmentation systems.

By analyzing the errors, we interestingly found that some of the erroneous splitting results are still acceptable to humans. For example, while 'アップロード (upload)' was annotated as a single word in the test data, our system split it into 'アップ (up)' and 'ロード (load)'. Although the latter splitting may be useful in some applications, it is judged as wrong in our evaluation framework. This implies the importance of evaluating the splitting results in some extrinsic tasks. We leave it to a future work.

6.4 Investigation on out-of-vocabulary words

In our test data, 2681 out of the 5286 katakana noun compounds contained at least one out-of-vocabulary word that are not registered in NAIST-jdic. Table 6 illustrates the results of the supervised systems for those 2681 and the remaining 2605 katakana noun compounds (referred to as w/ OOV and w/o OOV data, respectively). While the accuracy exceeds 90% for w/o OOV data, it is substantially degraded for w/ OOV data. This is consistent with our claim that out-of-vocabulary words are a major source of errors in splitting noun compounds.

The three supervised systems performed almost equally for w/o OOV data. This is because AP trivially performs very well on this subset, and it is difficult to get any further improvement. On the other hand, we can see that there are substantial performance gaps between the systems for w/ OOV data. This result reflects the effect of the additional fea-

[7] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html

[8] http://sourceforge.net/projects/mecab

Table 6: Splitting results of the supervised systems for w/ OOV and w/o OOV data.

System	w/ OOV data				w/o OOV data			
	P	**R**	**F_1**	**Acc**	**P**	**R**	**F_1**	**Acc**
AP	66.9	69.9	68.3	72.8	**95.4**	93.2	94.3	94.2
AP+GMF2	69.7	73.7	71.6	75.2	95.2	92.4	93.7	93.6
PROPOSED	**76.8**	**79.3**	**78.0**	**80.9**	95.3	**94.2**	**94.8**	**94.5**

tures more directly than is shown in table 5.

6.5 Effect of the two new features

To see the effect of the new features in more detail, we looked at the performances of our system using different feature sets (Table 7). The first column represents the feature set we used: BASIC, PARA, TRANS, and ALL represent the basic features, the paraphrase feature, the back-transliteration feature, and all the features. The results demonstrate that adding either of the new features improved the performance, and the best result was when they were used together. In all cases, the improvement over BASIC was statistically significant ($p < 0.01$, McNemar's test).

Next, we investigated the coverage of the features. Our test data comprised 7709 constituent words, 4937 (64.0%) of which were covered by NAIST-jdic. The coverage was significantly improved when using the back-transliteration feature. We observed that 6216 words (80.6%) are in NAIST-jdic or word-aligned transliteration pairs extracted from the Web text. This shows that the back-transliteration feature successfully reduced the number of out-of-vocabulary words. On the other hand, we observed that the paraphrase and back-transliteration features were activated for 79.5% (1926/2423) and 15.5% (376/2423) of the word boundaries in our test data.

Overall, we see that the coverage of these features is reasonably good, although there is still room for further improvement. It would be beneficial to use larger Web corpora or more paraphrase rules, for example, by having a system that automatically learns rules from the corpora (Barzilay and McKeown, 2001; Bannard and Callison-Burch, 2005).

6.6 Sensitivity on the threshold θ

Finally we investigated the influence of the threshold θ (Figure 1 and 2). Figure 1 illustrates the system performance in terms of F_1-score for different values of θ. While the F_1-score drops when the value of θ is too large (e.g., -20), the F_1-score is otherwise almost constant. This demonstrates it is generally easy to set θ near the optimal value. More importantly, the F_1-score is consistently higher than BASIC irrespective of the value of θ. Figure 2 represents the number of distinct word-aligned transliteration pairs that were extracted from the Web corpora. We see that most of the extracted transliteration pairs have high confidence score.

Table 7: Effectiveness of paraphrase (PARA) and back-transliteration feature (TRANS).

Feature set	P	R	F_1	Acc
BASIC	81.9	82.5	82.2	83.4
BASIC+PARA	85.1	85.3	85.2	85.9
BASIC+TRANS	85.1	86.3	85.7	86.5
ALL	**86.4**	**87.4**	**87.1**	**87.6**

7 Conclusion

In this paper, we explored the idea of using monolingual and bilingual paraphrases for splitting katakana noun compounds in Japanese. The experiments demonstrated that our method significantly improves the splitting accuracy by a large margin in comparison with the previously proposed methods. This means that paraphrasing provides a simple and effective way of using unlabeled textual data for identifying implicit word boundaries within katakana noun compounds.

Although our investigation was restricted to katakana noun compounds, one might expect that a similar approach would be useful for splitting other types of noun compounds (e.g., German noun compounds), or for identifying general word boundaries, not limited to those between nouns, in Asian languages. We think these are research directions worth exploring in the future.

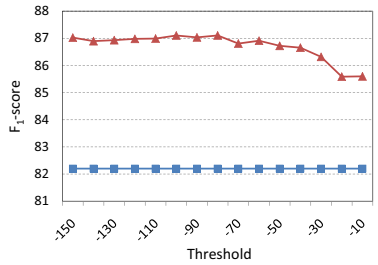


Figure 1: Influence of the threshold θ (x-axis) on the F_1-score (y-axis). The triangles and squares represent systems using the ALL and BASIC feature sets, respectively.

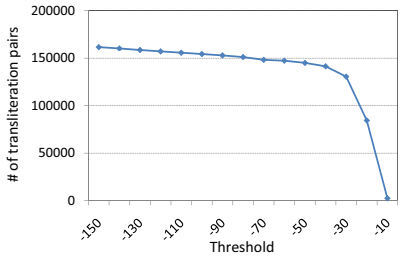


Figure 2: The number of distinct word-aligned transliterations pairs that were extracted from the Web corpora for different values of θ.

Acknowledgement

This work was supported by the *Multimedia Web Analysis Framework towards Development of Social Analysis Software* program of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008a. Decompoundig query keywords from compounding languages. In *Proceedings of ACL, Short Papers*, pages 253–256.

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008b. German decompounding in a difficult corpus. In *Proceedings of CICLing*, pages 128–139.

Rie Kubota Ando and Lillian Lee. 2003. Mostly-unsupervised statistical segmentation of Japanese Kanji sequences. *Natural Language Engineering*, 9(2):127–149.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*, pages 597–604.

Mohit Bansal and Dan Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of ACL*, pages 693–702.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*, pages 50–57.

Martin Braschler and Bärbel Ripplinger. 2004. How effective is stemming and decompounding for German text retrieval? *Information Retrieval*, 7:291–316.

Jamese Breen. 2009. Identification of neologisms in Japanese by corpus analysis. In *Proceedings of eLexicography in the 21st centry conference*, pages 13–22.

Eric Brill, Gray Kacmarcik, and Chris Brockett. 2001. Automatically harvesting katakana-English term pairs from search engine query logs. In *Proceedings of NLPRS*, pages 393–399.

Ralf D. Brown. 2002. Corpus-driven splitting of compound words. In *Proceedings of TMI*.

Guihong Cao, Jianfeng Gao, and Jian-Yun Nie. 2007. A system to mine large-scale bilingual dictionaries from monolingual Web pages. In *Proceedings of MT Summit*, pages 57–64.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of NAACL*, pages 406–414.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Florian Holz and Chris Biemann. 2008. Unsupervised and knowledge-free learning of compound splits and periphrases. In *CICLing*, pages 117–127.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignment and hidden Markov models to letter-to-phoneme conversion. In *HLT-NAACL*, pages 372–379.

Kyo Kageura, Fuyuki Yoshikane, and Takayuki Nozawa. 2004. Parallel bilingual paraphrase rules for noun compounds: Concepts and rules for exploring Web language resources. In *Proceedings of Workshop on Asian Language Resources*, pages 54–61.

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

Philip Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of EACL*, pages 187–193.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of EMNLP*, pages 230–237.

Sadao Kurohashi and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 22–38.

Toshiaki Nakazawa, Daisuke Kawahara, and Sadao Kurohashi. 2005. Automatic acquisition of basic Katakana lexicon from a given corpus. In *Proceedings of IJCNLP*, pages 682–693.

Preslav Nakov and Marti Hearst. 2005a. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of CoNLL*, pages 17–24.

Preslav Nakov and Marti Hearst. 2005b. Using the Web as an implicit training set: Application to structural ambiguity resolution. In *Proceedings of HLT/EMNLP*, pages 835–342.

Jong-Hoon Oh and Hitoshi Isahara. 2008. Hypothesis selection in machine transliteration: A Web mining approach. In *Proceedings of IJCNLP*, pages 233–240.

Naoaki Okazaki, Sophia Ananiadou, and Jin'ichi Tsujii. 2008. A discriminative alignment model for abbreviation recognition. In *Proceedings of COLING*, pages 657–664.

Anne Schiller. 2005. German compound analysis with wfsc. In *Proceedings of Finite State Methods and Natural Language Processing*, pages 239–246.

Ariel S. Schwartz and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of PSB*, pages 451–462.

Natsuko Tsujimura. 2006. *An Introduction to Japanese Linguistics*. Wiley-Blackwell.

Xianchao Wu, Naoaki Okazaki, and Jun'ichi Tsujii. 2009. Semi-supervised lexicon mining from parenthetical expressions in monolingual Web pages. In *Proceedings of NAACL*, pages 424–432.

Enhancing Chinese Word Segmentation Using Unlabeled Data

Weiwei Sun[†‡] and **Jia Xu**[‡]
[†]Department of Computational Linguistics, Saarland University
[‡]German Research Center for Artificial Intelligence (DFKI)
D-66123, Saarbrücken, Germany
wsun@coli.uni-saarland.de, Jia.Xu@dfki.de

Abstract

This paper investigates improving supervised word segmentation accuracy with unlabeled data. Both large-scale in-domain data and small-scale document text are considered. We present a unified solution to include features derived from unlabeled data to a discriminative learning model. For the large-scale data, we derive string statistics from Gigaword to assist a character-based segmenter. In addition, we introduce the idea about transductive, document-level segmentation, which is designed to improve the system recall for out-of-vocabulary (OOV) words which appear more than once inside a document. Novel features[1] result in relative error reductions of 13.8% and 15.4% in terms of F-score and the recall of OOV words respectively.

1 Introduction

Chinese sentences are written in continuous sequence of characters without explicit delimiters such as space characters. To find the basic language units, i.e. words, segmentation is a necessary initial step for Chinese language processing. Previous research shows that word segmentation models trained on labeled data are reasonably accurate. In this paper, we investigate improving supervised word segmentation with unlabeled data.

We distinguish three types of unlabeled data, namely large-scale in-domain data, out-of-domain data and small-scale document text. Both large-scale in-domain and out-of-domain data is popular for enhancing NLP tasks. Learning from these two types of unlabeled data normally involves semi-supervised learning. The difference between them is that out-of-domain data is usually used for domain adaptation. For a number of NLP tasks, there are relatively large amounts of labeled training data. In this situation, supervised learning can provide competitive results, and it is difficult to improve them any further by using extra unlabeled data. Chinese word segmentation is one of this kind of tasks, since several large-scale manually annotated corpora are publicly available. In this work, we first exploit unlabeled in-domain data to improve strong supervised models. We leave domain adaptation for our future work.

We introduce the third type of unlabeled data with a *transductive learning*, *document-level* view. Many applications of word segmentation involve processing a whole document, such as information retrieval. In this situation, the text of the current document can provide additional useful information to segment a sentence. Take the word "氨纶丝/elastane" for example[2]. As a translated terminology word, it lacks compositionality. Moreover, this word appears rarely in general texts. As a result, if it does not appear in the training data, it is very hard for statistical models to recognize this word. Nevertheless, when we deal with an article discussing an elastane company, this word may appear more than once in this article, and the document information can help recognize this word. This idea is closely related to transductive learning in the sense that the segmentation model knows something about the problem it

[1]You can download our derived features at http://www.coli.uni-saarland.de/~wsun/semi-cws-feats-emnlp11.tgz.

[2]This example is from an article indexed as chtb_0041 in the Penn Chinese Treebank corpus.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 970–979,
Edinburgh, Scotland, UK, July 27–31, 2011.

is going to resolve. In this work, we are also concerned with enhancing word segmentation with the document information.

We present a unified "feature engineering" approach for learning segmentation models from both labeled and unlabeled data. Our method is a simple two-stage process. First, we use unannotated corpus to extract string and document information, and then we use these information to construct new statistics-based and document-based feature mapping for a discriminative word segmenter. We are relying on the ability of discriminative learning method to identify and explore informative features, which play central role to boost the segmentation performance. This simple solution has been shown effective for named entity recognition (Miller et al., 2004) and dependency parsing (Koo et al., 2008). In their implementations, word clusters derived from unlabeled data are imported as features to discriminative learning approaches.

To demonstrate the effectiveness of our approach, we conduct experiments on the Penn Chinese Treebank (CTB) data. CTB is a collection of documents which are separately annotated. This annotation style allows us to evaluate our transductive segmentation method. Our experiments show that both statistics-based and document-based features are effective in the word segmentation application. In general, the use of unlabeled data can be motivated by two concerns: First, given a fixed amount of labeled data, we might wish to leverage unlabeled data to improve the performance of a supervised model. Second, given a fixed target performance level, we might wish to use unlabeled data to reduce the amount of annotated data necessary to reach this target. We show that our approach yields improvements for fixed data sets, even when large-scale labeled data is available. The new features result in relative error reductions of 13.8% and 15.4% in terms of the balanced F-score and the recall of out-of-vocabulary (OOV) words respectively. By conducting experiments on data sets of varying sizes, we demonstrate that for fixed levels of performance, the new features derived from unlabeled data can significantly reduce the need of labeled data.

The remaining part of the paper is organized as follows. Section 2 describes the details of our system, especially the design of the derived features.

B	Current character is the start of a word consisting of more than one character.
E	Current character is the end of a word consisting of more than one character.
I	Current character is a middle of a word consisting of more than two characters.
S	Current character is a word consisting of only one character.

Table 1: The start/end representation.

Section 3 presents experimental results and empirical analysis. Section 4 reviews the related work. Section 5 concludes the paper.

2 Method

2.1 Discriminative Character-based Word Segmentation

The Character-based approach is a dominant word segmentation solution for Chinese text processing. This approach treats word segmentation as a sequence tagging problem, assigning labels to the characters indicating whether a character locates at the beginning of, inside or at the end of a word. This character-by-character method was first proposed by (Xue, 2003), and a number of discriminative sequential learning algorithms have been exploited, including structured perceptron (Jiang et al., 2009), the Passive-Aggressive algorithm (Sun, 2010), conditional random fields (CRFs) (Tseng et al., 2005), and latent variable CRFs (Sun et al., 2009). In this work, we use the *Start/End* representation to express the position information of every character. Table 2.1 shows the meaning of each character label. For example, the target label representation of the book title "赵紫阳总理的秘密日记/The Secret Journal of Premier Zhao Ziyang" is as follows.

赵	紫	阳	总	理	的	秘	密	日	记
B	I	E	B	E	S	B	E	B	E

Key to our approach is to allow informative features derived from unlabeled data to assist the segmenter. In our experiments, we employed three different feature sets: a baseline feature set which draws upon "normal" information from labeled training data, a statistics-based feature set that uses statistical information derived from a large-scale in-domain corpus, and a document-based feature set

that uses information encoded in the surrounding text.

2.2 Baseline Features

In this work, to train a good traditional supervised segmenter, our baseline feature templates includes the ones described in (Sun et al., 2009; Sun, 2010). These features are divided into two types: character features and word type features. Note that the word type features are indicator functions that fire when the local character sequence matches a word uni-gram or bi-gram. Dictionary containing word uni-grams and bi-grams is collected from the training data. To conveniently illustrate, we denote a candidate character token c_i with a context $...c_{i-1}c_ic_{i+1}...$. We use $c_{[s:e]}$ to express a string that starts at the position s and ends at the position e. For example, $c_{[i:i+1]}$ expresses a character bi-gram c_ic_{i+1}. The character features are listed below.

- *Character uni-grams*: c_s $(i-3<s<i+3)$
- *Character bi-grams*: c_sc_{s+1} $(i-3<s<i+3)$
- Whether c_s and c_{s+1} are *identical*, for $i-2<s<i+2$.
- Whether c_s and c_{s+2} are *identical*, for $i-4<s<i+2$.

The word type features are listed as follows.

- The *identity* of the string $c_{[s:i]}$ $(i-6<s<i)$, if it matches a word from the list of uni-gram words;
- The *identity* of the string $c_{[i:e]}$ $(i<e<i+6)$, if it matches a word; multiple features could be generated.
- The *identity* of the bi-gram $c_{[s:i-1]}c_{[i:e]}$ $(i-6<s,e<i+6)$, if it matches a word bi-gram from the list of uni-gram words.
- The *identity* of the bi-gram $c_{[s:i]}c_{[i+1:e]}$ $(i-6<s,e<i+6)$, if it matches a word bi-gram; multiple features could be generated.

Idiom In linguistics, idioms are usually presumed to be figures of speech contradicting the principle of compositionality. As a result, it is very hard to recognize out-of-vocabulary idioms for word segmentation. Nonetheless, the lexicon of idioms can be taken as a close set, which helps resolve the problem well. In our previous work (Sun, 2011), we collect 12992 idioms from several free online Chinese dictionaries. This linguistic resource is publicly available[3]. In this paper, we use this idiom dictionary to derive the following feature.

- Does c_i locate at the beginning of, inside or at the end of an idiom? If the string $c_{[s:i]}$ $(s<i)$ matches an item from the idiom lexicon, the feature template receives a string value "E-IDIOM". Similarly, we can define when this feature ought to be set to "B-IDIOM" or "I-IDIOM". Note that all idioms are larger than one character, so there is no "S-IDIOM" feature here.

2.3 Statistics-based Features

In order to distill information from unlabeled data, we borrow ideas from some previous research on unsupervised word segmentation. The statistical information acquired from a relatively large amount of unlabeled data are designed as features correlated with the position where a character locates in a word token. These features are based on three widely used criteria.

2.3.1 Mutual Information

Empirical mutual information is widely used in NLP. Informally, mutual information compares the probability of observing x and y together with the probabilities of observing x and y independently. If there is a genuine association between x and y, the $I(x,y)=\log\frac{p(x,y)}{p(x)p(y)}$ should be greater than 0.

Some previous work claimed that the larger the mutual information between two consecutive strings, the higher the possibility of the two strings being combined together. We adopt this idea in our character-based segmentation model. The empirical mutual information between two character bi-grams is computed by counting how often they appear in the large-scale unlabeled corpus. Given a

[3] http://www.coli.uni-saarland.de/~wsun/idiom.txt.

Chinese character string $c_{[i-2:i+1]}$, the mutual information between substrings $c_{[i-2:i-1]}$ and $c_{[i:i+1]}$ is computed as:

$$MI(c_{[i-2:i-1]}, c_{[i:i+1]}) = \log \frac{p(c_{[i-2:i+1]})}{p(c_{[i-2:i-1]})p(c_{[i:i+1]})}$$

For each character c_i, we incorporate the MI of the character bi-grams into our model. They include,

- $MI(c_{[i-2:i-1]}, c_{[i:i+1]})$,
- $MI(c_{[i-1:i]}, c_{[i+1:i+2]})$.

2.3.2 Accessor Variety

When a string appears under different linguistic environments, it may carry a meaning. This principle is introduced as the *accessor variety* criterion for identifying meaningful Chinese words in (Feng et al., 2004). This criterion evaluates how independently a string is used, and thus how likely it is that the string can be a word. Given a string s, which consists of l ($l \geq 2$) characters, we define the *left accessor variety* of $L^l_{av}(s)$ as the number of distinct characters that precede s in a corpus. Similarly, the *right accessor variety* $R^l_{av}(s)$ is defined as the number of distinct characters that succeed s.

We first extract all strings whose length are between 2 and 4 from the unlabeled data, and calculate their accessor variety values. For each character c_i, we then incorporate the following information into our model,

- Accessor variety of strings with length 4: $L^4_{av}(c_{[i:i+3]})$, $L^4_{av}(c_{[i+1:i+4]})$, $R^4_{av}(c_{[i-3:i]})$, $R^4_{av}(c_{[i-4:i-1]})$;
- Accessor variety of strings with length 3: $L^3_{av}(c_{[i:i+2]})$, $L^3_{av}(c_{[i+1:i+3]})$, $R^3_{av}(c_{[i-2:i]})$, $R^3_{av}(c_{[i-3:i-1]})$;
- Accessor variety of strings with length 2: $L^2_{av}(c_{[i:i+1]})$, $L^2_{av}(c_{[i+1:i+2]})$, $R^2_{av}(c_{[i-1:i]})$, $R^2_{av}(c_{[i-2:i-1]})$.

2.3.3 Punctuation as Anchor Words

Punctuation marks are symbols that indicate the structure and organization of written language, as well as intonation and pauses to be observed when reading aloud. Punctuation marks can be taken as perfect word delimiters, and can be used as anchor words to harvest lexical knowledge. The preceding and succeeding strings of punctuations carry additional wordbreak information, since punctuations should be segmented as a word. Note that such information is biased because not all words can appear before or after punctuations. For example, punctuations can not be followed by particles, such as "了", "着" and "过" which are indicators of aspects. Nevertheless, our experiments will show this kind of information is still useful for word segmentation.

When a string appears many times preceding or succeeding punctuations, there tends to be wordbreaks succeeding or preceding that string. To utilize the wordbreak information provided by punctuations, we extract all strings with length $l (2 \leq l \leq 4)$ which precede or succeed punctuations in the unlabeled data. We define the *left punctuation variety* of $L^l_{pv}(s)$ as the number of times a punctuation precedes s in a corpus. Similarly, the *right punctuation variety* $R^l_{pv}(s)$ is defined as the number of how many times a punctuation succeeds s. These two variables evaluate how likely a string can be separated at its start or end positions.

We first gather all strings surrounding punctuations in the unlabeled data, and calculate their punctuation variety values. The length of each string is also restricted between 2 and 4. For each character c_i, we import the following information into our model,

- Punctuation variety of strings with length 4: $L^4_{pv}(c_{[i:i+3]})$, $R^4_{pv}(c_{[i-3:i]})$;
- Punctuation variety of strings with length 3: $L^3_{pv}(c_{[i:i+2]})$, $R^3_{pv}(c_{[i-2:i]})$;
- Punctuation variety of strings with length 2: $L^2_{pv}(c_{[i:i+1]})$, $R^2_{pv}(c_{[i-1:i]})$.

Punctuations can be viewed as *mark-up's* of Chinese text. Our motivation to use the punctuation information to assist a word segmenter is similiar to (Spitkovsky et al., 2010) in a way to explore "artificial" word (or phrase) break symbols. In their work, four common HTML tags are successfully used as raw phrase bracketings to improve unsupervised dependency parsing.

2.3.4 Binary or Numeric Features

The derived information introduced above is all expressed as real values. The natural way to incorporate these statistics into a discriminative learning model is to directly use them as numeric features. However, our experiments show that this simple choice does not work well. The reason is that these statistics actually behave non-linearly to predict character labels. For each type of statistics, one weight alone cannot capture the relation between its value and the possibility that a string forms a word. Instead, we represent these statistics as discrete features.

For the mutual information, this is done by rounding down decimal number. The integer part of each MI value is used as a string feature. For the accessor variety and punctuation variety information, since their values are integer, we can directly use them as string features. The accessor variety and punctuation variety could be very large, so we set thresholds to cut off large values to deal with the data sparse problem. Specially, if an accessor variety value is greater than 50, it is incorporated as a feature "> 50"; if the value is greater than 30 but not greater than 50, it is incorporated as a feature "$30 - 50$"; else the value is individually incorporated as a string feature. For example, if the left accessory variety of a character bi-gram $c_{[i:i+1]}$ is 29, the binary feature "$L^2_{av}(c_{[i:i+1]})$=29" will be set to 1, while other related binary features such as "$L^2_{av}(c_{[i:i+1]}) = 15$" or "$L^2_{av}(c_{[i:i+1]}) > 50$" will be set to 0. Similarly, we can discretize the punctuation variety features. However, we only set one threshold, 30, for this value. These thresholds can be tuned by using held-out data.

2.4 Document-based Features

It is meaningless to derive statistics of a document and use it for word segmentation, since most documents are relatively short, and values are statistically unreliable. Our experiments confirm this idea. Instead, we propose the following binary features which are based on the *string count* in the given document that is simply the number of times a given string appears in that document. For each character c_i, our document-based features include,

- Whether the string count of $c_{[s:i]}$ is equal to that of $c_{[s:i+1]}$ $(i - 3 \leq s \leq i)$. Multiple features are generated for different string length.

- Whether the string count of $c_{[i:e]}$ is equal to that of $c_{[i-1:e]}$ $(i \leq e \leq i + 3)$. Multiple features are generated for different string length.

The intuition is as follows. The string counts of $c_{[s:i]}$ and $c_{[s:i+1]}$ being equal means that when $c_{[s:i]}$ appears, it appears inside $c_{[s:i+1]}$. In this case, $c_{[s:i]}$ is not independently used in this document, and this feature suggests the segmenter not assign a "S" or "E" label to the character c_i. Similarly, the string counts of $c_{[i:e]}$ and $c_{[i-1:e]}$ being equal means $c_{[i:e]}$ is not independently used in this document, and this feature suggests segmenter not assign a "S" or "B" label to c_i. We do not directly use the string counts to prevent a bias towards longer documents.

3 Experiments

3.1 Setting

The SIGHAN Bakeoffs provide several large-scale labeled data for the research on Chinese word segmentation. Although these data sets are labeled on continuous run texts, they do not contain the document boundary information. CTB is a segmented, part-of-speech tagged, and fully bracketed corpus in the constituency formalism. It is also an popular data set to evaluate word segmentation methods, such as (Jiang et al., 2009; Sun, 2011). CTB is a collection of documents which are separately annotated. This annotation style allows us to calculate the so-called document-based features and to further evaluate our approach. In this paper, we use CTB 6.0 as our main corpus and define the training, development and test sets according to the Chinese sub-task of the CoNLL 2009 shared task[4]. Table 2 shows the statistics of our experimental settings.

Data set	# of sent.	# of words	# of char.
Training	22277	609060	1004266
Devel.	1763	49646	83710
Test	2557	73152	121008

Table 2: Training, development and test data on CTB 6.0

[4]We would like to thank Prof. Nianwen Xue for the help with the division of the data

Chinese Gigaword is a comprehensive archive of newswire text data that has been acquired over several years by the Linguistic Data Consortium (LDC). The large-scale unlabeled data we use in our experiments comes from the Chinese Gigaword (LDC2005T14). We choose the Mandarin news text, i.e. Xinhua newswire. This data covers all news published by Xinhua News Agency (the largest news agency in China) from 1991 to 2004, which contains over 473 million characters.

F-score is used as the accuracy measure. Define precision p as the percentage of words in the decoder output that are segmented correctly, and recall r as the percentage of gold standard output words that are correctly segmented by the decoder. The (balanced) F-score is $2pr/(p+r)$. We also report the recall of OOV words. Note that, all idioms in our extra idiom lexicon are added into the in-vocabulary word list.

CRFsuite (Okazaki, 2007) is an implementation of Conditional Random Fields (CRFs) (Lafferty et al., 2001) for labeling sequential data. It is a speed-oriented implementation, which is written in pure C. In our experiments, we use this toolkit to learn global linear models for segmentation. We use the stochastic gradient descent algorithm to resolve the optimization problem, and set default values for other learning parameters.

3.2 Main Results

Table 3 summarizes the segmentation results on the development data with different configurations, representing a few choices between baseline, statistics-based and document-based feature sets. In this table, the symbol "+" means features of current configuration contains both the baseline features and new features for semi-supervised or transductive learning. From this table, we can clearly see the impact of features derived from the large-scale unlabeled data and the current document. Comparison between the performance of the baseline and "+MI" shows that the widely used mutual information is not helpful. Both good segmentation techniques and valuable labeled corpora have been developed, and pure supervised systems can provide strong performance. It is not a trial to design new features to enhance supervised models.

There are significant increases when accessor variety features and punctuation variety features are separately added. Extending the length of neighboring string helps a little from 2 to 3. Although the OOV recall increases when the length is extended from 3 to 4, there is no improvement of the overall balanced F-score. The line "+MI+AV(2,3,4)+PU(2,3,4)" shows the performance when all statistics-based features are added. The combination of the "AV" and "PU" features gives further helps. This system can be seen as a pure semi-supervised system. The line "+DOC" is the result when document-based features are added. In spite of its simplicity, the document-based features can help the task. However, when we combine statistics-based features with document-based features, we cannot get further improvement in terms of F-score.

Devel.	P	R	$F_{\beta=1}$	R_{oov}
Baseline	95.41	95.52	95.46	77.68
+MI	95.50	95.48	95.49	77.98
+AV(2)	95.85	96.04	95.94	79.31
+AV(2,3)	95.95	96.19	96.07	80.61
+AV(2,3,4)	96.14	95.99	96.07	81.83
+PU(2)	95.86	96.07	95.97	79.70
+PU(2,3)	95.98	96.25	96.11	80.42
+PU(2,3,4)	96.00	96.19	96.10	80.53
+MI+AV(2,3,4)+PU(2,3,4)				
	96.17	96.22	96.19	80.42
+DOC	95.69	95.64	95.66	79.89
+MI+AV(2,3,4)+PU(2,3,4)+DOC				
	96.21	96.23	96.22	81.75

Table 3: Segmentation performance with different feature sets on the development data. Abbreviations: MI=mutual information; AV=accessor variety; PU=punctuation variety; DOC=document features. The numbers in each bracket pair are the lengths of strings. For example, PU(2,3) means punctuation variety features of character bi-grams and tri-grams are added.

Table 4 shows the segmentation performance on the test data set. The final results of our system are achieved with the "+MI+AV(2,3,4)+PU(2,3,4)+DOC" feature configuration. The new features result in relative error reductions of 13.8% and 15.4% in terms of the balanced F-score and the recall of OOV words respectively.

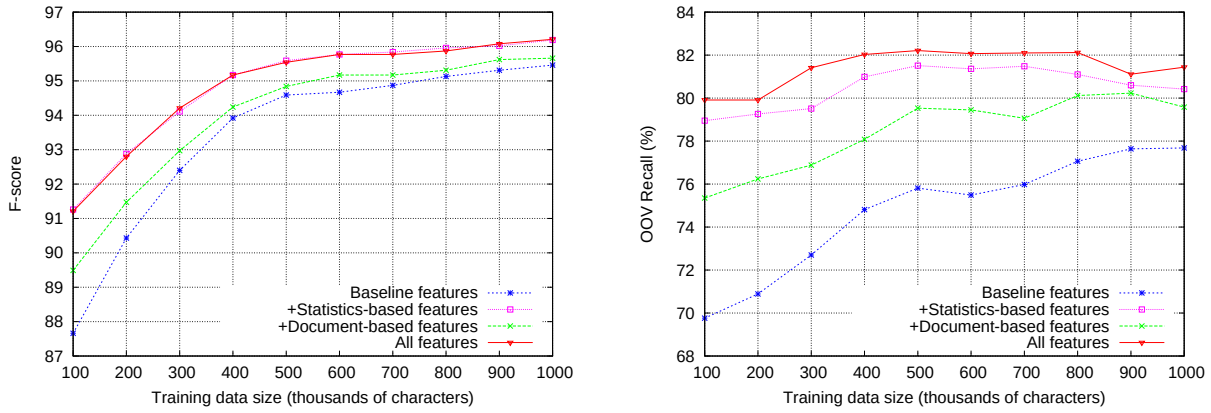


Figure 1: The learning curves of different models.

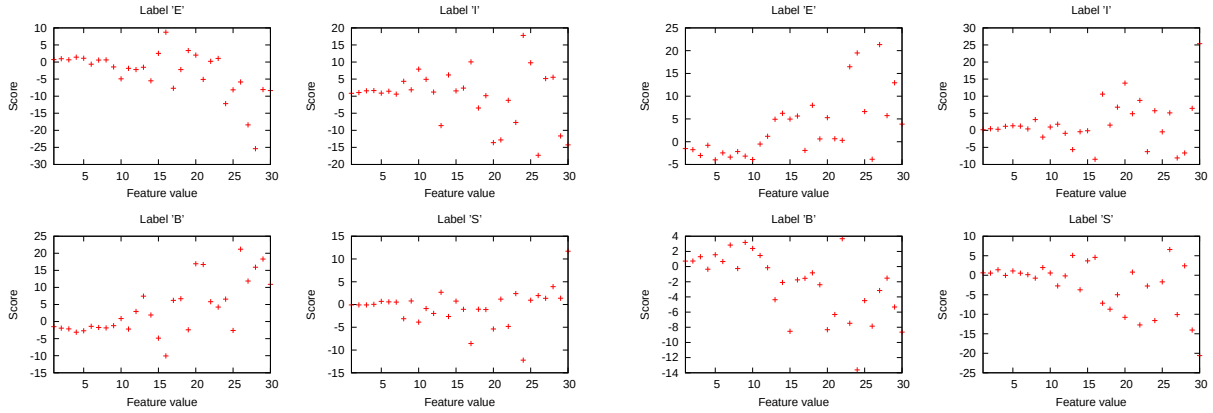


Figure 2: Scatter plot of feature score against feature value. The left side shows is $L^2_{pv}(c_{[i:i+1]}$ feature while the right side is the $R^2_{pv}(c_{[i:i+1]}$ feature.

Test	P	R	$F_{\beta=1}$	R_{oov}
Baseline	95.21	94.90	95.06	75.52
Final	95.86	95.62	95.74	79.28

Table 4: Segmentation performance on the test data.

3.3 Learning Curves

We performed additional experiments to evaluate the effect of the derived features as the amount of training data is varied. Figure 1 displays the F-score and the OOV recall of systems with different feature sets when trained on smaller portions of the labeled data. Note that there is no change in the configuration of the unlabeled data. We can clearly see that the derived features obtain consistent gains regardless of the size of the training set. The improvement is more significant when little labeled data is applied. Both statistics-based features and document-based features can help improve the overall performance. Especially, they can help to recognize more unknown words, which is important for many applications. The F-score of semi-supervised models, i.e. models trained with statistics-based features, does not achieve further improvement when document-based features are added. Nonetheless, the OOV recall obtains slightly improvements.

It is interesting to consider the amount by which derived features reduce the need for supervised data, given a desired level of accuracy. The change of the F-score in Figure 1 suggests that derived features reduce the need for supervised data by roughly a factor of 2. For example, the performance of the model with extra features trained on 500k characters

is slightly higher than the performance of the model with only baseline features trained on the whole labeled data.

3.4 Feature Analysis

We discussed the choice of using binary or numeric features in Section 2.3.4. In our experiment, when the accessor variety and punctuation variety information are integrated as numeric features, they do not contribute. To show the non-linear way that these features contribute to the prediction problem, we present the scatter plots of the score of each feature (i.e. the weight multiply the feature value) against the value of the feature. Figure 2 shows the relation between the score and the value of the punctuation variety features. For example, the weight of the binary feature "$L^2_{pu}(c_{[i:i+1]}) = 26$ combined with the label "B" learned by the final model is 0.815141, so the score of this combination is $0.815141 \times 26 = 21.193666$ and a point $(26, 21.193666)$ is drawn. These plots indicate the punctuation variety features contribute to the final model in a very complicated way. It is impossible to use one weight to capture it. The accessor variety features affect the model in the same way, so we do not give detailed discussions. We only show the same scatter plot of the $L^2_{av}(c_{[i:i+1]})$ feature template in Figure 3.

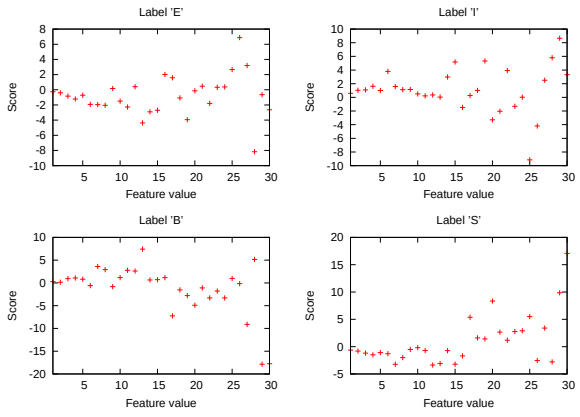


Figure 3: Scatter plot of feature score against feature value for $L^2_{av}(c_{[i:i+1]})$.

4 Related Work

Xu et al. (2008) presented a Bayesian semi-supervised approach to derive task-oriented word segmentation for machine translation (MT). This method learns new word types and word distributions on unlabeled data by considering segmentation as a hidden variable in MT. Different from their concern, our focus is general word segmentation.

The "feature-engineering" semi-supervised approach has been successfully applied to named entity recognition (Miller et al., 2004) and dependency parsing (Koo et al., 2008). These two papers demonstrated the effectiveness of using word clusters as features in discriminative learning. Moreover, Turian et al. (2010) compared different word clustering algorithms and evaluated their effect on both named entity recognition and text chunking.

As mentioned earlier, the feature design is inspired by some previous research on word segmentation. The accessor variety criterion is proposed to extract word types, i.e. the list of possible words, in (Feng et al., 2004). Different from their work, our method resolves the segmentation problem of running texts, in which this criterion is used to define features correlated with the character position labels. Li and Sun (2009) observed that punctuations are perfect delimiters which provide useful information for segmentation. Their method can be viewed as a self-training procedure, in which extra punctuation information is incorporated to filter out automatically predicted samples. We use the punctuation information in a different way. In our method, the counts of the preceding and succeeding strings of punctuations are incorporated directly as features into a supervised model.

In machine learning, transductive learning is a learning framework that typically makes use of unlabeled data. The goal of transductive learning is to only infer labels for the unlabeled data points in the test set rather than to learn a general classification function that can be applied to any future data sets. This means that the test data is known as a priori knowledge and can be used to construct better hypotheses. Although the idea to explore the document-level information in our work is similar to transductive learning, we do not use state-of-the-art transductive learning algorithms which involve learning when they meet the test data. For real-world applications, our approach is efficient by avoiding re-training.

5 Conclusion and Future Work

In this paper, we have presented a simple yet effective approach to explore unlabeled data for Chinese word segmentation. We are concerned with large-scale in-domain data and the document text. Experiments show that our approach achieves substantial improvement over a competitive baseline. Especially, the informative features derived from unlabeled data lead to significant improvement of the recall of unknown words. Our immediate concern for future work is to exploit the out-of-domain data to improve the robustness of current word segmentation systems. The idea would be to extract domain information from unlabeled data and define them as features in our unified approach. The word-based approach is an alternative for word segmentation. This kind of segmenters sequentially predicts whether the local sequence of characters make up a word. A natural avenue for future work is the extension of our method to the word-based approach. The word segmentation task is similar to constituency parsing, in the sense of finding boundaries of language units. Another interesting question is whether our method can be adapted to resolve constituency parsing.

Acknowledgments

The work is supported by the project TAKE (Technologies for Advanced Knowledge Extraction), funded under contract 01IW08003 by the German Federal Ministry of Education and Research. The author is also funded by German Academic Exchange Service (DAAD).

References

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor variety criteria for Chinese word extraction. *Comput. Linguist.*, 30:75–93.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging – a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 522–530. Association for Computational Linguistics, Suntec, Singapore.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603. Association for Computational Linguistics, Columbus, Ohio.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for Chinese word segmentation. *Comput. Linguist.*, 35:505–512.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 337–342. Association for Computational Linguistics, Boston, Massachusetts, USA.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010. Profiting from mark-up: Hypertext annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1278–1287. Association for Computational Linguistics, Uppsala, Sweden.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Coling 2010: Posters*, pages 1211–1219. Coling 2010 Organizing Committee, Beijing, China.

Weiwei Sun. 2011. A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of the ACL 2011 Conference*. Association for Computational Linguistics, Portland, Oregon, United States.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A

discriminative latent variable Chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64. Association for Computational Linguistics, Boulder, Colorado.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *In Fourth SIGHAN Workshop on Chinese Language Processing*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, Uppsala, Sweden.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised Chinese word segmentation for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1017–1024. Coling 2008 Organizing Committee, Manchester, UK.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*.

Unsupervised Learning of Selectional Restrictions and Detection of Argument Coercions

Kirk Roberts and Sanda M. Harabagiu
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083, USA
{kirk,sanda}@hlt.utdallas.edu

Abstract

Metonymic language is a pervasive phenomenon. Metonymic type shifting, or argument *type coercion*, results in a selectional restriction violation where the argument's semantic class differs from the class the predicate expects. In this paper we present an unsupervised method that learns the selectional restriction of arguments and enables the detection of argument coercion. This method also generates an enhanced probabilistic resolution of logical metonymies. The experimental results indicate substantial improvements the detection of coercions and the ranking of metonymic interpretations.

1 Introduction

Metonymic language is pervasive in today's social interactions. For example, it is typical to find questions that require metonymic resolution:

(Q1) Did you enjoy War and Peace?
(Q2) Does anyone have any advice on how to start a bowling team?[1]

In order to process such questions and capture the intention of the person that posed them, coercions are needed. Question (Q1) is interpreted as whether you enjoyed *reading* "War and Peace", while (Q2) is interpreted as asking for advice on *organizing*, *forming*, or *registering* a bowling team. The quality of the answers therefore depends on the ability to (1) recognize when metonymic language is used, and (2) to produce coercions that capture the user's intention. One important step in this direction was taken by SemEval-2010 Task 7, which focused on the ability to recognize (a) an argument's selectional restriction for predicates such as *arrive at*, *cancel*, or *hear*, and (b) the type of coercion that licensed a correct interpretation of the metonymy. Details of the task are reported in (Pustejovsky et al., 2010). Approaches to metonymy based on this task are limited, however, because (a) the task is focused only on semantically non-ambiguous predicates and (b) the selectional restrictions of the arguments were chosen from a pre-defined set of six semantic classes (artifact, document, event, location, proposition, and sound). However, metonymy coercion systems capable of providing the interpretations of questions (Q1) and (Q2) clearly cannot operate with the simplifications designed for this task.

Inspired by recent advances in modeling selectional preferences with latent-variable models (Ritter et al., 2010; Ó Séaghdha, 2010), we propose an unsupervised model for learning selectional restrictions. The model assumes that (1) arguments have a single selected class exemplified by the selectional restriction, and (2) the selected class can be inferred from the data, in part by modeling how coercive each predicate is. The model is capable of operating with both ambiguous and disambiguated predicates, producing superior results for predicates that have been disambiguated. The selectional restrictions and coercions detected by the model reported in this paper can be used to enhance the logical metonymy approach reported in Lapata and Lascarides (2003). The experimental results show a significant improvement in the ranking of interpretations.

[1] Both questions taken from Yahoo Answers.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 980–990,
Edinburgh, Scotland, UK, July 27–31, 2011.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 details unsupervised models that inform detection of metonymies. Section 4 outlines a method for disambiguating ambiguous predicates. Section 5 describes the enhanced interpretation of logical metonymies when conventional constraints are known. Section 6 outlines our implementation and experimental design. Section 7 presents our experimental results in three broad tasks: (i) semantic class induction, (ii) coercion detection, and (iii) logical metonymy interpretation. Section 8 summarizes the conclusions.

2 Previous Work

Lapata and Lascarides (2003) propose a probabilistic ranking model for logical metonymies. They estimate these probabilities using co-occurrence frequencies of predicate-argument pairs in a corpus. Shutova (2009) extends this approach to provide sense-disambiguated interpretations from WordNet (Fellbaum, 1998) by using the alternative interpretations to disambiguate polysemous words. Shutova and Teufel (2009) extend this approach further by clustering these sense-disambiguated interpretations into distinct groups of meaning (e.g., {*read*, *browse*, *look through*} and {*write*, *produce*, *work on*} for "*enjoy book*"). Not only do these approaches assume logical metonymies have already been identified, but they are susceptible to providing interpretations that are themselves logical metonymies (e.g., *finish book*). In this paper, we propose an enhancement to resolving logical metonymies by ruling out event-invoking predicates in order to provide more semantically valid interpretations.

Recently, the resolution of several linguistic problems has benefited from Latent Dirichlet Allocation (LDA) (Blei et al., 2003) models. Ó Séaghdha (2010) examines several selectional preference models based on LDA in predicting human judgements on predicate-argument plausibility. Both LDA and an extension, ROOTH-LDA (based on Rooth et al. (1999)), perform well at predicting plausibility on unseen predicate-argument pairs. Inspired by these results, we propose to extend selectional preference models in order to learn selectional restrictions.

Alternatively, unsupervised algorithms exist that both induce semantic classes (Rooth et al., 1999; Lin and Pantel, 2001) and cluster predicates by their selectional restrictions (Rumshisky et al., 2007) but none of these provide a sufficient framework for determining if a specific argument violates its predicate's selectional restriction.

3 Unsupervised Learning of Selectional Restrictions

In predicate-argument structures, predicates impose selectional restrictions in the form of semantic expectations on their arguments. Whenever the semantic class of the argument meets these constraints a *selection* occurs. For example, the predicate "*hear*" imposes the semantics related to sound on the argument "*voice*". Because the semantic class for "*voice*" conforms to these constraints, we call its semantic class the *selected class*. However, when the semantic class of the argument violates these constraints, we follow Pustejovsky et al. (2010) and refer to this as a *coercion*. In this case, we call the argument's semantic class the *coerced class*. For example, "*hear speaker*" is a coercion where the argument class, person, is implicitly coerced into the voice of the speaker, a sound.

3.1 A Baseline Model

We consider the LDA-based selectional preference model reported in Ó Séaghdha (2010) as a baseline for modeling selectional restrictions. Formally, we define our LDA baseline model as follows. Let V be the predicate vocabulary size, let A be the argument vocabulary size, and let K be the number of argument classes. Let a_i^v be the i^{th} (non-unique) argument realized by predicate v. Let c_i^v be the class for a_i^v. Let θ^v be the class distribution for predicate v and ϕ^k be the argument distribution for class k. The graphical model for this LDA is shown in Figure 1(a). The generative process for LDA is:

For each argument class $k = 1..K$:
1. Choose $\phi^k \sim$ Dirichlet(β)

For each unique predicate $v = 1..V$:
2. Choose $\theta^v \sim$ Dirichlet(α)

For every argument $i = 1..n^v$:
3. Choose $c_i^v \sim$ Multinomial(θ^v)
4. Choose $a_i^v \sim$ Multinomial($\phi^{c_i^v}$)

Following Griffiths and Steyvers (2004), we collapse θ and ϕ and estimate the model using Gibbs

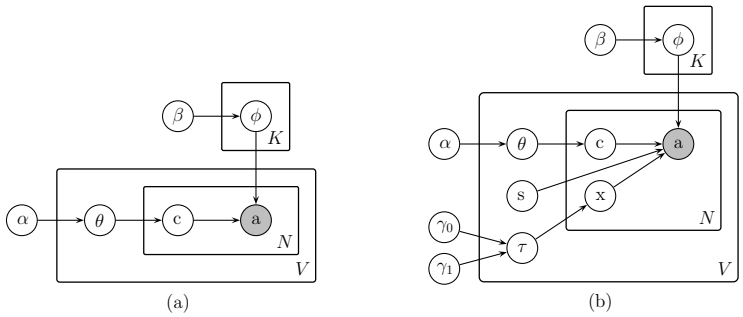


Figure 1: Graphical models for (a) LDA, and (b) coercion LDA (cLDA).

Sampling. This yields the update equation:

$$p(c_i^v = k|\mathbf{a}^v; \alpha, \beta) \propto \frac{f_{vk} + \alpha}{f_v + K\alpha} \frac{f_{ak} + \beta}{f_k + A\beta} \quad (1)$$

Where f_{ak} is the frequency of argument a being assigned class k; f_k is the frequency of class k being assigned to any argument; f_{vk} is the frequency of predicate v having an argument of class k; and f_v is the total number of non-unique arguments for predicate v.

3.2 A Coercion Model

We now incorporate our assumptions for selectional restriction modeling. Namely: (1) there is one selected class per predicate, and (2) the predicate's selected class can be chosen from the classes of its arguments. To accomplish this, we must also account for the coerciveness of each predicate. We assign a latent variable τ^v for each predicate v that controls how coercive v should be. The additional hyper-parameters γ_0 and γ_1 act as priors on τ^v. The generative process for this coercion LDA model, which we denote cLDA, is:

For each argument class $k = 1..K$:
1. Choose $\phi^k \sim$ Dirichlet(β)

For each unique predicate $v = 1..V$:
2. Choose $s^v \sim$ Uniform(1,K)
3. Choose $\theta^v \sim$ Dirichlet(α)[2]
4. Choose $\tau^v \sim$ Beta(γ_0, γ_1)

For every argument $i = 1..n^v$:
5. Choose $c_i^v \sim$ Multinomial(θ^v)
6. Choose $x_i^v \sim$ Bernoulli(τ^v)
7. If $x_i^v = 1$, Choose $a_i^v \sim$ Multinomial($\phi^{c_i^v}$)
   Else Choose $a_i^v \sim$ Multinomial(ϕ^{s^v})

The model variable s^v represents the selected class for predicate v. The coerced class is represented for each argument i by c_i^v, where x_i^v chooses between the selected and coerced class. The variable x_i^v is similar to switching variables in other graphical models such as Chemudugunta et al. (2007) and Reisinger and Mooney (2010), where switching variables are used to choose between a background distribution and a document-specific distribution. In this case, the switching variable chooses between a specific class and a predicate-specific distribution. The graphical model for cLDA is shown in Figure 1(b). Note that cLDA is virtually equivalent to LDA when τ^v is 1 and γ_1 is small because the selected class will be ignored. In this way, highly coercive predicates have less of an impact on the argument clustering because they are more reliant on the multinomial θ. We use Gibbs sampling to perform model inference and collapse θ, ϕ, and τ, integrating them out using multinomial-Dirichlet conjugacy (the Beta distribution used by τ is just a special case of the Dirichlet with only two parameters).

The update formula for the selected class s^v is:

$$\begin{aligned} p(s^v = k|\mathbf{a}^v, \mathbf{c}^v, \mathbf{x}^v; \alpha, \beta) \\ &\propto \prod_i^{n^v} P(a_i^v|s^v = k; \beta) \\ &\propto \prod_{i \in S^v} \frac{f_{a_i^v k} + \beta}{f_k + A\beta} \end{aligned} \quad (2)$$

Where n^v is the number of argument observations for predicate v; S^v is the set of arguments of v that are selections; and $f_{a_i^v k}$ is the frequency of word a_i^v being assigned to class k for any predicate. We then sample c_i^v and x_i^v jointly:

$$\begin{aligned} p(c_i^v = k, x_i^v = q|s^v, \mathbf{c}_{\bar{i}}^v, \mathbf{x}_{\bar{i}}^v, \mathbf{a}^v; \alpha, \beta, \gamma) \\ &\propto p(c_i^v{=}k; \alpha) p(x_i^v{=}q; \gamma) p(a_i^v|s^v, \mathbf{c}_{\bar{i}}^v, \mathbf{x}_{\bar{i}}^v, \mathbf{a}^v; \beta) \\ &\propto \frac{f_{vk} + \alpha}{f_v + K\alpha} \frac{f_{vq} + \gamma_q}{f_{v0} + \gamma_0 + f_{v1} + \gamma_1} \frac{f_{az} + \beta}{f_z + A\beta} \end{aligned} \quad (3)$$

[2]With the exception that the probability of drawing the selected class s^v is zero. This can be seen as drawing the multinomial θ^v from a Dirichlet distribution with K-1 components.

Where f_{vq}, f_{v0}, and f_{v1} is the frequency of x values that equal q, 0, and 1, respectively, for predicate v; f_{az} is the frequency of word a being in class z and f_z is the frequency all words being in class z, where z is defined as being equal to k when $x_i^v = 1$, or s^v when $x_i^v = 0$.

Note that Equation (2) results in a sampling of the selected class for v proportional to the number of arguments in each class for v, fulfilling our second assumption. Also note from Equation (3), the second term corresponds to the coerciveness of the predicate. When the predicate is very coercive, the marginal probability associated with $x_i^v = 0$ will be very low. If all predicates become entirely coercive, most x values will become 1 and the cLDA will become almost equivalent to an LDA model.

3.3 Coercion Detection

After the latent parameters have been estimated, we still require a method to determine if a given predicate-argument pair is a coercion or not. We assign a score in $[0, 1]$ instead of a binary value. Higher scores (near 1) indicate high likelihood of selection, while lower scores (near 0) indicate coercion. The LDA model must rely on a scoring method using the predicate-class and argument-class mixtures:

$$C_1(v, a) = \sum_k^K P(k|v)P(a|k) = \sum_k^K \theta_k^v \phi_a^k \quad (4)$$

Where θ_k^v represents the probability of any argument of v being in the class k and ϕ_a^k represents the probability of the argument a being in class k for any predicate. C_1 is also available as a scoring method for cLDA by including the proportion of the selected class s^v in θ. Note that since θ and ϕ are integrated out for both LDA and cLDA, we instead use their frequencies smoothed with α and β, respectively, which is their maximum likelihood estimate.

The cLDA model contains two useful parameters that can identify selections and coercions: the selected class s and the coercion indicator x. This yields two more coercion scoring metrics:

$$C_2(v, a) = P(a|s^v) = \phi_a^{s^v} \quad (5)$$

$$C_3(v, a) = P(x_a^v = 0|v, a) = 1.0 - \frac{\sum_{i \in I_a^v} x_i^v}{|I_a^v|} \quad (6)$$

Where s^v is the selected class for predicate v; I_a^v is the set of predicate-argument instances for predicate v and argument a; and x_i^v is 0 for a selection and 1 for a coercion. Of the three metrics, C_3 is the most direct measure of a coercion as it represents the average decision the model learned on the same predicate-argument pair. However, C_3 requires a large sample of instances for a particular predicate and argument, and so may be quite sparse. In practice, these different metrics have their own strengths and weaknesses and the best performing method often depends on the final task.

4 Predicate Sense Induction

Our assumption of a single selected class per predicate ignores predicate polysemy. However, the same lexical item may have multiple meanings, each with a separate selected class. We therefore propose a method of partitioning a predicate's arguments by the induced senses of the predicate. This allows separate induced predicates to each select a separate argument class. Consider the verb *fire*, which has at least two distinct common senses: (1) to shoot or propel an object (e.g., to fire a gun), and (2) to lay someone off (e.g., to fire an employee). The first sense selects a weapon (e.g., gun, bullet, rocket), while the second sense selects a person (e.g., employee, coach, apprentice).

Specifically, we employ tiered clustering (Reisinger and Mooney, 2010) using the words in the predicate's context. Tiered clustering is a discrete clustering method, as opposed to methods such as (Brody and Lapata, 2009) that assign a distribution of word senses to each word instance. Tiered clustering has several advantages over other discrete clustering approaches. First, tiered clustering learns a background word distribution in addition to the clusters. This reduces the impact that words common to most senses have on the clustering process and allow clusters to form around only the most salient words. Second, tiered clustering

Cluster 1 (18,391)	Cluster 2 (16,651)	Cluster 3 (18,749)	Cluster 4 (11,833)
shots	ball	hire	gun
gun	puck	letter	imagination
Israeli	hired	Yeltsin	grill
missiles	owner	minister	laser
rockets	shots	workforce	cells
officers	coaches	executives	engine
soldiers	net	employee	brain
rounds	circle	managers	!
bullets	Johnson	hired	engines
weapons	Williams	union	fire

Table 1: Context word clusters resulting from tiered clustering for the verb *fire* (includes the number of unique words belonging to each cluster).

uses a Chinese Restaurant Process (CRP) prior to control both the formation of new clusters (senses) and the bias toward larger clusters (more common senses). This conforms with our intuition of how word senses are distributed: a few common senses with a gradual transition to a long tail of rare senses. When deciding which cluster to use for a given predicate-argument pair, we use the cluster most associated with the argument.

We use a 10-token window around the predicate as features. The result of predicate induction on the verb *fire* is shown in Table 4. The first three clusters can be interpreted to be about (1) firing weapons, (2) sport-related shots (e.g., "*fired the puck*"), and (3) lay-offs. One must be careful in choosing the parameters for induction, however, as it is possible to partition a unique word sense such that coercions and selections are placed in a separate clusters. Section 6 discusses our parameter selection experiments.

5 Logical Metonymy Interpretation

Logical metonymies are a unique class of coercions due to the fact that their eventive interpretation can be derived from verbal predicates. For instance, for the logical metonymy "*enjoy book*", we know that *read* is a good candidate interpretation because (1) books are objects whose purpose is to be read and (2) reading is an event that may be enjoyed. We therefore expect to see many instances of both "*read book*" and "*enjoy reading*" (Lapata and Lascarides, 2003). Conversely, for coercions with non-eventive interpretations, such as "*arrive at meeting*", the interpretation (*location of*) is more dependent on the predicate (*arrive*) than the function of its argument (*meeting*).

In this section, we limit our discussion of logical metonymy to the verb-object case, its corresponding baseline for ranking interpretations, and our proposed enhancements. However, similar baselines exist for other types of logical metonymy, such as adjective-noun and noun-noun. Since our enhancement does not depend on any syntactic information beyond the predicate-argument instances needed for Section 3.2, it could easily be applied to those as well.

Lapata and Lascarides (2003) propose a probabilistic ranking model where the probability of an interpretation e for a verb-object pair (v, o) is proportional to the probability of all three in a verb-interpretation-object pattern.[3] For example, the probability that *read* is the correct interpretation of "*enjoy book*" is proportional to the likelihood of seeing "*enjoy reading book*" expressed as a syntactic dependency in a sufficiently large corpus. Due to data sparsity, they approximate this likelihood of seeing the object given the verb and interpretation to simply the likelihood of seeing the object given the interpretation. We denote this logical metonymy ranking method as LM_{LL}, formally defined as:

$$\begin{aligned} LM_{LL}(e; v, o) &= P_c(v, e, o) \\ &= P_c(e)P_c(v|e)P_c(o|e, v) \\ &\approx P_c(e)P_c(v|e)P_c(o|e) \\ &\approx \frac{f_c(v, e)f_c(o, e)}{N f_c(e)} \end{aligned} \quad (7)$$

Where P_c and f_c indicate probability and frequency, respectively, derived from corpus counts. See Lapata and Lascarides (2003) for a detailed explanation of how these frequencies are obtained.

This model, which we consider our baseline, is only partially correct as the corpus will contain coercions that form invalid interpretations. Consider the phrases "*enjoy finishing a book*" and "*enjoy discussing a book*". Both "*finish book*" and "*discuss book*" are coercions (and logical metonymies) themselves, and do not form a valid interpretation.[4]

[3] They use two patterns: "*v e*-ing *o*" and "v to e *o*", where e is tagged as a verb.

[4] For evidence of the frequency of these phrases, at the time of this writing, "*enjoy finishing a book*" and "*enjoy finishing the book*" have a combined 728 Google hits, while "*enjoy discussing a book*" and "*enjoy discussing the book*" have a com-

Thus, when discovering interpretations for logical metonymies, we must be aware of the selectional restrictions of candidate interpretations.

We propose to incorporate the coercion probability learned by our cLDA model in order to rank only those interpretations that are considered selections:

$$LM'(e; v, o) = P(v, e, o, x_o^e = 0) \quad (8)$$

However, due to the approximations made to estimate $P_c(v, e, o)$, this probability cannot be directly calculated as not all the frequencies reflect verb-object counts. Instead, we can combine the corpus probability $P_c(v, e, o)$ with the probability that the verb-object pair (e, o) is a coercion in our model. We denote this probability $P_x(e, o)$, and it may be derived from the scoring metrics in Equations (4), (5), or (6) above. We further propose three methods for enhancing the LM_{LL} baseline using $P_x(e, o)$ to approximate Equation 8.

A naive method for including information from our cLDA model is to consider the corpus probability, $P_c(v, e, o)$ and the coercion probability, $P_x(e, o)$, to be independent:

$$LM_{IND}(e; v, o) = P_c(v, e, o)P_x(e, o) \quad (9)$$

In other words, the rank of an interpretation is dictated by the unweighted combination of its corpus probability P_c and its coercion probability P_x. However, these two quantities are not likely to be independent. Most instances where e is used with either v or o are in fact selective.[5] We therefore experiment with two shallow learning methods for combining these two quantities.

The first method is a filtering approach where a threshold is learned for P_x:

$$LM_{TH}(e; v, o) = \begin{cases} P_c(v, e, o) & \text{if } P_x(e, o) \geq \delta \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

Where the threshold δ is learned from a development set. We expect this model could suffer from noisy P_x values or to simply choose a threshold of zero due to the prominence of P_c.

Finally, we include a weighted linear model to discover the relative value of P_c and P_x:

$$LM_{WT}(e; v, o) = w_1 P_c(v, e, o) + w_2 P_x(e, o) \quad (11)$$

Where w_1 and w_2 are learned weights. We discuss how the parameters for LM_{TH} and LM_{WT} are learned in the experimental setup below.

6 Experimental Setup

We use the NYT subsection of the English Gigaword Fourth Edition (Parker et al., 2009) for a total of 1.8M newswire articles. The Stanford Dependency Parser (de Marneffe et al., 2006) is used to extract verb-object relations (*dobj*) that form the input to our model. To reduce noise, we keep only verbs listed in VerbNet (Kipper et al., 1998) with at least 100 argument instances, discarding *have* and *say*, which are too semantically flexible to select from clear semantic classes and so common they distort the class distributions. This results in 4,145 unique verbs with 51M argument instances (388K unique arguments). Additionally we use the dependency parser to extract open clausal complements of verbs (e.g., "*like to swim*") for use in logical metonymy interpretation. We believe this to be a more reliable alternative to the phrase chunk extraction patterns used in Lapata and Lascarides (2003). We keep clausal complements (*xcomp*) where the dependent is either a gerund or infinitive in order to estimate $P_c(v|e)$ in Equation (7).

For tiered clustering we use the same implementation as Reisinger and Mooney (2010)[6] to partition the surface form of the verb into one or more induced forms. Instead of using a fixed number of iterations, the clustering was run for 100 iterations past the best recorded log-likelihood in order to find the best possible fit to the data. We tuned the hyperparameters by maximizing the log-likelihood on a small held-out set of 20 predicate-argument pairs (10 selections, 10 coercions). The resulting partitions were fairly conservative, yielding 12,332 induced verbs or about 3 induced verb forms for every surface form, with 305 verbs not being partitioned at all.

We implemented both LDA and cLDA as described in Sections 3.1 and 3.2. For the α and β

bined 7,040 Google hits.

[5] For comparison, "*enjoy reading a book*" and "*enjoy reading the book*" have a combined 6.5 million Google hits

[6] Available at http://github.com/joeraii/UTML-Latent-Variable-Modeling-Toolkit

hyper-parameters, we used the MALLET (McCallum, 2002) defaults of 1.0 and 0.1, respectively, for both LDA and cLDA. We used the 20 predicate-argument pairs mentioned above to tune the γ hyper-parameters as well as the number of iterations. Both γ_0 and γ_1 were set to 100. We observed that for both LDA and cLDA, longer runs (in iterations) resulted in improved model log-likelihood but inferior results in terms of detecting coercions. It is not uncommon in topic modeling for model likelihood to not be completely correlated with the score on the task for which the topic model was intended (see Chang et al. (2009)). Both LDA and cLDA were found to perform best at 50 iterations on this data, after which their class distributions were less "smooth" and became rigidly associated with just a few classes, thus having a negative impact on coercion detection. While further iterations hurt coercion detection, only minor gains in model likelihood are seen. We believe the small number of iterations necessary for the model to converge is therefore a function of the data. In traditional topic modeling, documents are generally of similar size (i.e., within an order of magnitude). But in our data, many predicates have 10,000 times more instances than others. We have not yet empirically explored the impact of using a more uniform number of arguments for each predicate. This issue also makes it difficult to take multiple samples, which we experimented with unsuccessfully.

Our a priori intuition was that as the number of classes was increased, LDA would improve and cLDA would degrade due to its assumption of a single selected class. However, this did not always bear out in the results for every task described below. As such, instead of choosing a specific number of classes for each model, we describe results for each model with $K = 10$, 25, and 50.

For logical metonymy, both LM_{TH} and LM_{WT} require learned parameters. LM_{TH} needs a learned threshold while LM_{WT} needs two learned weights. For both, we split the data set into two partitions, learn the optimal threshold/weights on one partition, and use it as the parameters for the other partition. Both methods are trained on the final scoring metric, described in Section 7.3. For threshold learning, this involves finding the optimal cut-off to maximize the score. For weight learning, we use an exhaustive search over the range {1.0, 0.9, ..., 0.2, 0.1, 10^{-2}, 10^{-3}, ..., 10^{-14}} for both w_1 and w_2.

induced predicates?		N			Y		
# classes		10	25	50	10	25	50
LDA	NMI	.382	.448	.389	.435	.391	.383
	Rand	.717	.731	.721	.760	.723	.730
	F1	.425	.319	.192	.543	.311	.205
	B^3 (C)	.553	.513	.444	.525	.476	.341
	B^3 (E)	.453	.351	.223	.521	.324	.234
	MUC	.545	.545	.531	.500	.532	.544
cLDA	NMI	.446	.403	.360	**.510**	.430	.366
	Rand	.736	.719	.716	**.788**	.734	.711
	F1	.448	.291	.183	**.567**	.329	.184
	B^3 (C)	.575	.484	.312	**.593**	.495	.313
	B^3 (E)	.473	.321	.205	**.556**	.346	.205
	MUC	.500	.521	.507	**.595**	.541	.571

Table 2: Clustering scores for induced classes.

7 Results and Discussion

7.1 Semantic Class Induction

For the evaluation of the argument classes induced by our method, we use a subset of the WordNet lexicographer files, which correspond to coarse-grained semantic classes. We chose this form of evaluation because, unlike a named entity corpus, no sentential context is required and is therefore more consistent with the information available to our model. We use six of the larger, more semantically coherent WordNet classes: artifact, person, plant, animal, location, and food. We consider each of these a cluster and compare them to clusters composed of the top ten non-polysemous words (according to WordNet) in each of the classes generated by both the baseline (LDA) and our model (cLDA). Words not in both sets of clusters are removed. The result of this evaluation, compared with six clustering metrics, is shown in Table 2. For descriptions of NMI, Rand, and cluster F-measure, see Manning et al. (2008); for the B^3 metrics (Cluster and Element), see Bagga and Baldwin (1998); for the MUC metric, see Vilain et al. (1995). Each metric has different strengths and biases in regards to the number and distribution of clusters, so all are provided to give a general picture of class induction performance.

The best performing model on all metrics is cLDA with induced predicates using 10 classes. However, as the number of classes is increased and the granularity of the induced classes becomes more fine-grained, LDA (predictably) outperforms cLDA on most metrics. This is consistent with our intuition

induced predicates?		N			Y		
# classes		10	25	50	10	25	50
LDA	C_1	74.4	78.7	80.5	69.7	70.1	73.4
cLDA	C_1	80.6	**81.2**	80.9	76.2	78.4	77.5
	C_2	75.4	75.9	78.9	73.5	68.3	**80.8**
	C_3	67.8	70.8	67.4	70.9	67.4	74.1

Table 3: Accuracy on SemEval-2010 Task 7 data.

that a single-class assumption degrades as the number of classes increases.

For this evaluation, predicate induction also improved LDA for smaller numbers of classes, but not to the degree that it improved cLDA. Without predicate induction, LDA outperforms cLDA on all six metrics for 25 and 50 classes. With predicate induction, LDA outperforms cLDA on only one metric for 25 classes and five metrics for 50 classes. Thus the induced predicates do reduce the negative impact caused by the single selected class assumption for semantic class induction.

7.2 Coercion Detection

For the evaluation of coercion detection, we use the SemEval-2010 Task 7 data (Pustejovsky et al., 2010). This data uses the most common sense for each of five predicates (*arrive*, *cancel*, *deny*, *finish*, and *hear*) with a total of 2,070 sentences annotated with the argument's source type (the argument's semantic class) and target type (the predicate's selected class for that argument). We ignore the actual argument classes and evaluate on the coercion type, which is a selection when the source and target type match, and a coercion otherwise.

In order to evaluate unsupervised systems on this data, we use the corresponding training set (1,031 examples) to learn a threshold for coercion detection. At test time, if the model output is below the threshold, a coercion is inferred. Otherwise it is considered a selection. Therefore, the better a model can rank selections over coercions, the more accurate threshold it will learn. The results for this evaluation are shown in Table 3. The baseline for this task (threshold = 0, or all selections) is 67.4.

The best overall model on this data is cLDA using the C_1 coercion scoring method (Equation (4)). This method consistently outperforms the baseline LDA, especially for smaller numbers of classes, performing best with $K = 25$. The second metric, C_2, was not as reliable. The third metric, C_3, performed poorly on the task. As discussed in Section 3.3, C_3 is a direct result of the sampling for the predicate-argument pair in question and can thus be expected to perform poorly on rare predicate-argument pairs. Given that many of the arguments in this data are rare or unseen in the Gigaword data (e.g., "*cancel Renault*"), C_3's poor performance is understandable.

The use of predicate sense induction based on tiered clustering to overcome the single-class assumption caused significant degradation in performance on this task. Using automatically induced predicates instead of the surface form caused an average degradation of 2.6 points across the twelve tests. A potential explanation for this is that the evaluated predicates have a single dominant sense, meaning the single class assumption may be valid for these predicates (the task-defined selected classes are: location for *arrive*, event for *cancel* and *finish*, proposition for *deny*, and sound for *hear*). Therefore it would be interesting to evaluate it on a set of highly polysemous predicates with multiple dominant senses. Furthermore, the introduction of predicate sense induction was designed to help cLDA, and the performance degradation for these nine tests was not as large as it was for LDA. For cLDA, C_1 had an average degradation of 3.5 points compared to LDA's C_1 average degradation of 6.5 points. cLDA's C_2 had an average degradation of only 2.5 points and C_3 was actually improved by 2.1 points. This suggests that there is value in assigning different selected classes via sense induction, but that the two-step approach is not beneficial for these common predicates. This could be overcome by a joint approach of inducing predicate classes while simultaneously detecting coercions, as the presence of many coercions would be an indicator that more induced predicates are necessary.

7.3 Logical Metonymy Interpretation

For the evaluation of logical metonymy, we use both an existing data set and a newly created data set. Shutova and Teufel (2009) annotated 10 verb-object logical metonymies from Lapata and Lascarides (2003) with sense-disambiguated interpretations and organized the interpretations into clusters representing different possible meanings. For evaluation purposes we ignore the sense annotations and clusters and consider all lexical matchings of one of the annotated interpretations to be correct. The

	induced predicates?		N			Y		
	# classes		10	25	50	10	25	50
LM_{LL}			0.381			0.365		
LM_{IND}	LDA	C_1	0.415	0.406	0.383	0.386	0.412	0.395
	cLDA	C_1	0.408	0.412	0.412	0.407	**0.468**	0.439
		C_2	0.415	0.447	0.419	0.414	0.415	0.434
		C_3	0.416	0.453	**0.455**	0.395	0.416	0.402
LM_{TH}	LDA	C_1	0.599	0.568	0.588	0.479	0.520	0.551
	cLDA	C_1	0.571	0.644	0.751	0.497	0.620	**0.708**
		C_2	0.544	0.496	0.633	0.457	0.635	0.660
		C_3	0.601	0.677	**0.767**	0.472	0.622	0.571
LM_{WT}	LDA	C_1	0.383	0.381	0.379	0.365	0.356	0.361
	cLDA	C_1	0.380	0.387	0.381	0.386	0.377	0.321
		C_2	0.317	0.342	0.350	0.338	0.340	0.345
		C_3	0.378	0.370	0.350	0.387	0.382	0.384

Table 4: Mean average precision (MAP) scores on the Shutova and Teufel (2009) data set. The bold items indicate the best scores with/without induced predicates as well as using/not using a threshold-based interpretation method.

	induced predicates?		N			Y		
	# classes		10	25	50	10	25	50
LM_{LL}			0.274			0.248		
LM_{IND}	LDA	C_1	0.291	0.286	0.294	0.263	0.267	0.255
	cLDA	C_1	0.296	0.298	0.285	0.280	0.274	0.288
		C_2	0.291	0.287	0.288	0.283	0.271	0.285
		C_3	0.318	0.317	**0.333**	0.298	0.285	**0.307**
LM_{TH}	LDA	C_1	0.478	0.534	0.534	0.414	0.495	0.479
	cLDA	C_1	0.449	0.504	0.541	0.391	0.495	**0.513**
		C_2	0.505	0.478	0.456	0.398	0.429	0.440
		C_3	0.449	0.496	**0.577**	0.382	0.439	0.446
LM_{WT}	LDA	C_1	0.276	0.270	0.271	0.248	0.251	0.249
	cLDA	C_1	0.271	0.272	0.270	0.257	0.259	0.265
		C_2	0.274	0.274	0.266	0.250	0.259	0.261
		C_3	0.271	0.273	0.274	0.253	0.262	0.259

Table 5: Mean average precision (MAP) scores on 100 logical metonymies manually annotated with interpretations. The bold items indicate the best scores with/without induced predicates as well as using/not using a threshold-based interpretation method.

data contains an average of 11 interpretations per metonymy and has a reported 70% recall.

In order to create a larger data set, we identified 100 verb-object logical metonymies, including those used in Lapata and Lascarides (2003). Three annotators were asked to provide up to five interpretations for each metonymy (they were not provided with any verbs from which to choose, only the verb-object pair). The annotators provided an average of 4.6 interpretations per metonymy. Because our goal was recall, inter-annotator agreement was necessarily low, and each logical metonymy had an average of 11.7 unique interpretations. All annotators agreed on at least one interpretation for 40 metonymies, while for 14 they had no interpretations in common.[7]

Since logical metonymy interpretation is usually evaluated as a ranking task, we score our methods using mean average precision (MAP):

$$\text{MAP} = \frac{1}{Q}\sum_{q=1}^{Q}\frac{\sum_{n=1}^{N}\text{prec}(n)\times\text{rel}(n)}{\text{interps}(q)} \qquad (12)$$

Where Q is the number of metonymies evaluated; N is the number of interpretations ranked; prec(n) is the precision at rank n; rel(n) = 1 if interpretation n is valid, 0 otherwise; and interps(q) is the number of valid interpretations for the metonymy q. We rank all 4,145 verbs as interpretations except for those removed by the threshold technique, as they have a score of zero. This can give LM_{TH} artificially high MAP scores since it may remove some valid interpretations that are low-ranking. However, since a smaller, higher precision list may be useful for many applications we still consider MAP a valid metric and indicate both the highest scoring method and the highest scoring non-threshold method. The results on the Shutova and Teufel (2009) data are

[7] Data available at http://www.hlt.utdallas.edu/∼kirk/data/lmet.zip

shown in Table 4. The results on our own data are shown in Table 5.

The scores reported in the Shutova and Teufel (2009) data are noticeably higher than the data we annotated. Since the metonymies in our data are a super-set of those in their data, and since for those metonymies our annotators provided approximately the same number of interpretations (110 versus 120), this likely indicates the remaining metonymies in our data are more difficult.

In all cases the best reported scores use cLDA. Unlike coercion detection on the SemEval data, C_3 performs very well, achieving the highest scores when no predicate sense induction is used. Also unlike coercion detection, LDA scores do not increase as the number of classes increase. We suspect both these differences have to do with the fact that the arguments in this data are far more common. Since LDA is a selectional preference model and its coercion scores correspond roughly to the plausibility of seeing a predicate-argument pair, it is less able to distinguish coercions in common arguments.

Of the logical metonymy ranking methods, LM_{TH} consistently produces the highest MAP scores. However, as stated before, by using a cut-off and removing low-ranking valid interpretations, the MAP score is increased, which might not be applicable to some applications. The best non-thresholded ranking method is LM_{IND}, which naively combines the LM_{LL} score with the coercion probability. In almost every case this beats out LM_{WT}. Upon inspection, we observed that the range of scale for the LM_{LL} scores are very inconsistent. This can make it difficult to learn a linear model using these scores as features, and as a result the learned weights were forced to ignore the coercion score and rely entirely on LM_{LL}. We attempted other scaling methods, such as a rank-based method, but these had poor results as well, so we leave the problem of the supervised learning these weights to future work.

Using induced senses did not result in the drastic and consistent degradation in performance seen on the SemEval data, and the highest non-threshold result for the Shutova and Teufel (2009) data used predicate induction. Both metonymy data sets were limited to the verbs found in Lapata and Lascarides (2003), which are still quite common (*attempt*, *begin*, *enjoy*, *expect*, *finish*, *prefer*, *start*, *survive*, *try*, *want*). However, the verbs used in our data set had a greater number of WordNet senses attested in a corpus than the SemEval data (an average of 4.4 senses for our data versus 3.0 senses for the SemEval data). This suggests the potential value of sense induction for highly polysemous predicates and further motivates the integration of sense induction within a selectional restriction model.

8 Conclusion

We have presented a novel topic model that extends an unsupervised selectional preference model (LDA) to an unsupervised selectional restriction model (cLDA) using two assumptions. For the first assumption, that each predicate has a single selected class, we proposed a predicate induction method to overcome predicate polysemy. This improved results for semantic class induction but proved harmful for detecting coercions on common predicates with a single, dominant sense. For the second assumption, that the selected class can be inferred from the data, we proposed a sampling method based on the classes of the predicate's arguments. Superior performance on coercion detection shows the merit of this assumption.

Additionally, we proposed methods for improving an existing task, logical metonymy interpretation, using the learned parameters of our model, showing positive results.

It is clear that our model may be improved by more accurate predicate sense induction. To this end, we plan to develop a model that simultaneously induces predicates and learns coercions, using knowledge of a predicate's coerciveness to inform the induction mechanism.

Acknowledgements

We would like to thank Diarmuid Ó Séaghdha, Bryan Rink, and Anna Rumshisky for several helpful conversations during the course of this work. We thank Mirella Lapata and Ekaterina Shutova for making the data from their experiments available as well as the organizers of SemEval-2010 Task 7 for the associated data set. Additionally, we thank Srikanth Gullapalli, Aileen McDermott, and Bryan Rink for annotating the data set used in our experiment. Finally, we thank the anonymous reviewers for their suggestions on improving this work.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In Proceedings of the First International Conference on Language Resources and Evaluation Workshop on Linguistic Coreference*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 103–111.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems*, pages 1–9.

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2007. Modeling General and Specific Aspects of Documents with a Probabilistic Topic Model. In *Advances in Neural Information Processing Systems 19*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the Fifth International Language Resources and Evaluation*.

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

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 1998. Class-based construction of a verb lexicon. In *Proceedings of AAAI/IAAI*.

Maria Lapata and Alex Lascarides. 2003. A Probabilistic Account of Logical Metonymy. *Computational Linguistics*, 21(2):261–315.

Dekang Lin and Patrick Pantel. 2001. Induction of Semantic Classes from Natural Language Text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 317–322.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit.

Diarmuid Ó Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2009. English Gigaword Fourth Edition. *The LDC Corpus Catalog.*, LDC2009T13.

James Pustejovsky, Anna Rumshisky, Alex Plotnick, Elisabetta Jezek, Olga Batiukova, and Valeria Quochi. 2010. SemEval-2010 Task 7: Argument Selection and Coercion. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 27–32.

Joseph Reisinger and Raymond J. Mooney. 2010. A Mixture Model with Sharing for Lexical Semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A Latent Dirichlet Allocation method for Selectional Preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

Anna Rumshisky, Victor A. Grinberg, and James Pustejovsky. 2007. Detecting selectional behavior of complex types in text. In *Fourth International Workshop on Generative Approaches to the Lexicon*.

Ekaterina Shutova and Simone Teufel. 2009. Logical Metonymy: Discovering Classes of Meaning. In *Proceedings of the CogSci 2009 Workshop on Semantic Space Models*.

Ekaterina Shutova. 2009. Sense-based interpretation of logical metonymy using a statistical method. In *Proceedings of the ACL 2009 Student Workshop*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings fo the 6th Message Understanding Conference*, pages 45–52.

Harnessing different knowledge sources to measure semantic relatedness under a uniform model

Ziqi Zhang **Anna Lisa Gentile** **Fabio Ciravegna**
Department of Computer Science, University of Sheffield
211 Portobello, Regent Court
Sheffield, S1 4DP
z.zhang@dcs.shef.ac.uk a.l.gentile@dcs.shef.ac.uk f.ciravegna@dcs.shef.ac.uk

Abstract

Measuring semantic relatedness between words or concepts is a crucial process to many Natural Language Processing tasks. Exiting methods exploit semantic evidence from a single knowledge source, and are predominantly evaluated only in the general domain. This paper introduces a method of harnessing different knowledge sources under a uniform model for measuring semantic relatedness between words or concepts. Using Wikipedia and WordNet as examples, and evaluated in both the general and biomedical domains, it successfully combines strengths from both knowledge sources and outperforms state-of-the-art on many datasets.

1 Introduction

Semantic relatedness (SR) measures how much two (strings of) words or concepts are related by encompassing all kinds of relations between them (Strube and Ponzetto, 2006). It is more general than semantic similarity. SR is often an important pre-processing step to many complex Natural Language Processing (NLP) tasks, such as Word Sense Disambiguation (Leacock and Chodorow, 1998; Han and Zhao, 2010), and information retrieval (Finkelstein et al., 2002). In the biomedical domain, SR is an important technique for discovering gene functions and interactions (Wu et al., 2005; Ye et al., 2005).

There is an abundant literature on measuring SR between words or concepts. Typically, these methods extract semantic evidence of words and concepts from a background knowledge source, with which their relatedness is assessed. The knowledge sources can be unstructured documents or (semi-)structured resources such as Wikipedia, WordNet, and domain specific ontologies (e.g., the Gene Ontology[1]).

In this paper, we identify two issues that have not been addressed in the previous works. First, existing works typically employ a single knowledge source of semantic evidence. Research (Strube and Ponzetto, 2006; Zesch and Gurevych, 2010; Zhang et al., 2010) has shown that the accuracy of an SR method differs depending on the choice of the knowledge sources, and there is no conclusion which knowledge source is superior to others. Zhang et al. (2010) argue that this indicates different knowledge sources may complement each other. Second, the majority of SR methods have been evaluated in general domains only, except a few earlier WordNet-based methods that have been adapted to biomedical ontologies and evaluated in that domain (Lord et al., 2003; Pedersen et al., 2006; Pozo et al., 2008). Given the significant attention that SR has received in specific domains (Pesquita et al., 2007), evaluation of SR methods in specific domains is increasingly important.

This paper addresses these issues by proposing a generic and uniform model for computing SR between words or concepts using multiple knowledge sources, and evaluating the proposed method in both general and specific domains. The method combines and integrates semantic evidence of words or concepts extracted from any knowledge source in a generic graph representation, with which the SR between concepts or words is computed. Using two of the most popular general-domain knowledge sources,

[1] http://www.geneontology.org/, last retrieved in Mar. 2011

Wikipedia and WordNet as examples, the method is evaluated on 7 benchmarking datasets, including three datasets from the biomedical domain and four from the general domain. It has achieved excellent results: compared to the baselines that use each single knowledge sources, combining both knowledge sources has improved the accuracy on all datasets by 2~11%; compared to state-of-the-art on the general domain datasets, the method achieves the best results on three datasets; and on the other three biomedical datasets, it obtains the best result in one case; and second and third best results on the other two among eight participating methods, where all other competitors exploit some domain-specific knowledge sources.

The remainder of this paper is organized as follows. Section 2 discusses related work; Section 3 presents the proposed method; Section 4 describes the experiments and evaluation; Section 5 discusses results and findings; Section 6 concludes this paper.

2 Related work

2.1 SR methods

Methods for computing SR can be classified into *path based*, *Information Content (IC) based*, *statistical* and *hybrid methods*. *Path based* methods (Hirst and St-Onge, 1998; Leacock and Chodorow, 1998; Pekar and Staab, 2002; Rada et al., 1989; Wu and Palmer, 1994) measure SR between words or concepts as a function of their distance in a semantic network, usually calculated based on the path connecting the words or concepts by certain semantic (typically *is-a*) links. *IC based methods* (Jiang and Conrath, 1997; Lin, 1998; Pirro et al., 2009; Resnik, 1995; Seco et al., 2004) assess relatedness between words or concepts by the amount of information they share, usually determined by a higher level concept that subsumes both concepts in a taxonomic structure. *Statistical methods* measure relatedness between words or concepts based on their distribution of contextual evidence. This can be formalized as co-occurrence statistics collected from unstructured documents (Chen et al., 2006; Cilibrasi and Vitanyi, 2007; Matsuo et al., 2006), or distributional concept or word vectors with features extracted from either unstructured documents (Harrington, 2010; Wojtinnek and Pulman, 2011) or (semi-)structured knowledge resources (Agirre et al., 2009; Gabrilovich and Markovitch, 2007; Gouws et al., 2010; Zesch and Gurevych, 2007; Zhang et al., 2010). *Hybrid methods* combine different purebred methods in certain ways. For example Riensche et al. (2007) employ both an *IC based* method (Resnik, 1995) and a *statistical* method (cosine vector similarity) in their study. Pozo et al. (2008) derive a taxonomy of terms from unstructured documents by applying hierarchical clustering based on corpus statistics, then apply *path based* method on this taxonomy to compute SR. Han and Zhao (2010) use one *IC based* method and two *statistical* methods to compute SR, then derive an aggregated score.

2.2 SR knowledge sources and domains

Computing SR requires background knowledge about concepts or words, which can be extracted from unstructured corpora, semi-structured and structured knowledge resources. Unstructured corpora are easier to create and cheaper to maintain, however, semantic relations between words or concepts are implicit. Methods (Chen et al., 2006; Cilibrasi and Vitanyi 2007; Matsuo et al., 2006) that exploit unstructured corpora typically depend on distributional statistics, and thus may ignore important semantic evidences present in (semi-)structured knowledge sources (Pan and Farrell, 2007). Recent studies (Harrington, 2010; Pozo et al., 2008; Wojtinnek and Pulman, 2011) propose to pre-process a corpus to learn a semantic network, with which SR is computed. This creates high pre-processing cost; also, the choice of corpus and its size often have a direct correlation with the accuracy of SR methods (Batet et al., 2010).

(Semi-)Structured knowledge sources on the other hand, organize semantic knowledge about concepts and words explicitly and interlink them with semantic relations. They have been popular choices in the studies of SR, and they include lexical resources such as WordNet, Wiktionary, and (semi-)structured encyclopedic resources such as Wikipedia. WordNet has been used in earlier studies (Hirst and St-Onge, 1998; Jiang and Conrath, 1997; Lin, 1998; Leacock and Chodorow 1998; Resnik, 1995; Seco et al., 2004; Wu and Palmer, 1994) and is still a preferred knowledge source in recent works (Agirre et al., 2009). However, its effectiveness may be hindered by its lack of coverage of specialized lexicons and domain specific concepts (Strube and Ponzetto,

2006; Zhang et al., 2010). Wikipedia and Wiktionary are collaboratively maintained knowledge sources and therefore may overcome this limitation. Wikipedia in particular, is found to have reasonable coverage of many domains (Holloway et al., 2007; Halavais, 2008). It has become increasingly popular in SR studies recently. However, research (Zesch and Gurevych, 2010) have shown that methods based on Wikipedia have no clear advantage over WordNet-based methods on some general domain datasets in terms of accuracy, while Zhang et al. (2010) argue that different knowledge sources may complement each other, and SR methods may benefit from harnessing different knowledge sources.

Several studies (Lord et al., 2003; Pedersen et al., 2006; Petrakis et al., 2006; Pozo et al., 2008) have adapted state-of-the-art to domain specific knowledge sources (e.g., the Gene Ontology, the MeSH[2]) and evaluated them therein. Despite these efforts, a large proportion of state-of-the-art is still only evaluated in the general domain.

2.3 SR methods similar to this work

Few works have attempted at combining different knowledge sources in SR studies, especially (semi-)structured knowledge sources. The closest studies are Han and Zhao (2010) and Tsang and Stevenson (2010). Han and Zhao firstly compute SR between words using three state-of-the-art SR methods separately. Next, one score is chosen subject to an arbitrary preference order, and used to create a connected graph of weighted edges between words. A recursive function is then applied to the graph to compute final SR scores between words. Essentially, each SR method is applied in isolation and features from different sources are used separately with each distinctive method. Although this retains advantages of each method, the limitations of them are also combined.

Tsang and Stevenson (2010) combine WordNet and unstructured documents by weighing each word found in WordNet using its frequency observed in a large corpus. The frequencies however, are sensitive to the choice of corpus, thus different corpora may result in different accuracies. Furthermore, their method is only applicable to computing SR between pairs of sets of words or concepts.

3 Methodology

We define a set of requirements for SR methods that harness different knowledge sources:

- It should improve over the same method based on a single knowledge source
- It should be generic and applicable to any knowledge source
- It should be robust in dealing with knowledge source specific features but also tolerate the quality and coverage issues of individual knowledge source

Our method of harnessing different knowledge sources contains four steps. Firstly (Section 3.1), each word or word segment is searched in each knowledge source to identify their *contexts* that is specific to that knowledge source. We define a *context* as the representation of meaning or a concept for a word. In the following, we say that each *context* is associated with a distinct concept. Secondly (Section 3.2), for each concept of an input word, features are extracted from its *context* and a graph representation of each concept and their features is created. Thirdly (Section 3.3), cross-source *contexts* are mapped where they refer to the same concept, thus their features from different sources can be combined to derive an enriched representation. This creates a final, uniform graph representation where input words are connected by shared features of their underlying candidate concepts. Then (Section 3.4) the graph is submitted to a generic algorithm to compute SR between words.

In the following, we discuss details with respect to different types of knowledge sources, while focusing on Wikipedia and WordNet in our experiments for two reasons. First, they are used by the majority of SR methods and are therefore most representative knowledge sources. Second, they have strongly distinctive and complementary characteristics, which make ideal testbeds for the requirements. On one hand, WordNet is a lexical resource containing rich and strict semantic relations between words, but lacks coverage of specialized vocabularies. On the other hand, Wikipedia is a semi-structured resource with good coverage of domains and named entities, but the semantic knowledge is organized in a looser way.

[2] http://www.nlm.nih.gov/mesh/ last retrieved in March 2011

3.1 Context retrieval

Given a pair of words or word segments, we firstly identify *contexts* representing the underlying meanings or concepts from each knowledge source. For lexical resources, this could be distinctive word senses. In **WordNet (WN)**, a context corresponds to a single synset, which corresponds to a concept. We search each word in WordNet and extract all possible synsets. Let *w* be a word or word segment (e.g., "cat"), and $C_w^{wn} = \{c_{1_w}^{wn}, c_{2_w}^{wn} \dots c_{k_w}^{wn}\}$ be the set of *k* concepts of *w* extracted from WordNet.

Using **Wikipedia (WK)** as an example semi-structured resource, the *context* can be an article that describes a unique concept. Thus we search for underlying articles that describe different concepts. Firstly, we search *w* in Wikipedia, where three situations may be anticipated. If a single non-disambiguation page describing a concept is returned, the concept is selected and the retrieval is complete. In the second case, a disambiguation page linking to all possible concept pages may be returned. This page lists all underlying concepts and entities referenced by *w* as links and a short description with each link. In this case, we always keep the first concept page, which is found often to be the most common sense of the word; additionally, we select other concept pages whose short descriptions contain the word *w*. We do not select all linked pages because many of these in fact link to a concept relevant to *w*, but not necessarily a candidate sense of *w*. Thirdly, if no pages are returned for *w*, we search for the most relevant page using *w* as keyword(s) in an inverted index of all Wikipedia pages (e.g., via search engines). We denote concepts retrieved from Wikipedia as $C_w^{wk} = \{c_{1_w}^{wk}, c_{2_w}^{wk} \dots c_{k_w}^{wk}\}$.

For unstructured sources such as documents, a simple approach could be defining a word *context* as a text passage around each occurrence of *w*, and grouping similar *contexts* of *w* as representation of its underlying meanings, or concepts. Alternatively, more complex approaches such as Pozo et al. (2008) and Harrington (2010) may be applied to extract a lexical network of words, whereby similar methods to WordNet can be applied.

3.2 Feature extraction and representation

Next, for each concept identified from a knowledge source, features are extracted from their corresponding *contexts*. In our case, for each $c \in C_w^{wk}$, we follow the work by Zhang et al. (2010) to extract four types of features from their corresponding Wikipedia pages. Figure 1 shows an example representation of a concept and its Wikipedia features:

- Words from page titles and redirection links (can be considered as synonyms)
- Words from categories, used as higher level hypernyms in some studies (Zesch et al., 2010; Strube and Ponzetto, 2006)
- Words from outgoing links
- Top *n* most frequent words from a page

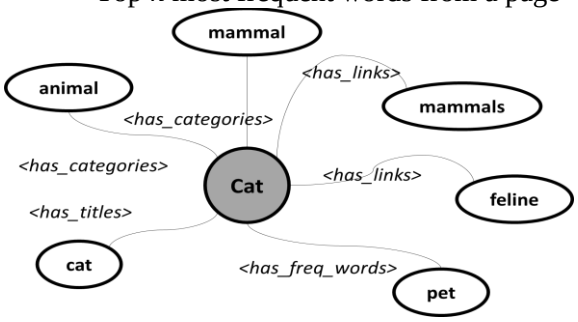


Figure 1. Representation of the concept "cat, the mammal" using different types of features extracted from Wikipedia. The shaded circle represents the concept; ovals represent feature values; edges connecting feature values to the concept and <labels> represent feature types

For each $c \in C_w^{wn}$, we extract ten features from WordNet: hypernyms, hyponyms, meronyms, holonyms, synonyms, antonyms, attributes, "see also" words, "related" words, and gloss. These are also represented in the same way as in Figure 1.

With unstructured sources, contextual words can be used as features. Alternatively, if a lexical network is extracted, features may be extracted in a similar way to those of WordNet.

Additionally, with WordNet and Wikipedia, we also propose several intra-resource feature merging strategies to study the effect of **feature diversification**. This is because, while some approaches (such as Agirre et al., 2009; Harrington, 2010; Yeh et al., 2009) do not distinguish different feature types in graph construction, or adopt a bag-of-words feature representation (such as Zesch and Gurevych, 2010), others (such as Yazdani and Popescu-Belis, 2010; Zhang et al., 2010) have used differentiated

feature types and weights in their model. We therefore carry out studies to investigate this issue. Specifically, for the original four Wikipedia features, we create a bag-of-words feature that simply merges all feature types (i.e., all edges in Figure 1 will have the same label). For the original ten WordNet features, we propose two merged representations corresponding to that of Wikipedia, so as to support the studies of feature enrichment in the following section. We introduce a bag-of-words feature that collapses all different feature types, and a four-feature representation as follow:

- *wn-synant* merges WordNet synonyms and antonyms.
- *wn-hypoer* merges WordNet hypernyms and hyponyms, collectively representing features by "*is-a*" semantic relation
- *wn-assc* merges WordNet meronyms, holonyms, related and "see also", which are features corresponding to associative relations
- *wn-dist* merges WordNet gloss and attributes that generally describe a concept.

3.3 Concept mapping and feature enrichment

Our method essentially harnesses different knowledge sources by combining features extracted from different sources in a uniform model. This requires two sub-processes: **cross-source concept mapping** and **cross-source feature enrichment**.

In **cross-source concept mapping,** concepts extracted from different knowledge sources are mapped according to similar meanings such that cross-source features can be combined. To do so, we select the concepts from one knowledge source as the reference concept set; then concepts from other knowledge sources are mapped to reference concepts of similar meanings. There can be different criteria of choosing reference knowledge source concepts. Empirically, we found it necessary to choose the knowledge source with broader coverage and richer features. This will be discussed later in Section 5. Following this strategy, in our example, C_w^{wk} is chosen as reference concepts, and for each $c_w^{wk} \in C_w^{wk}$ we select a $c_w^{wn} \in C_w^{wn}$ such that c_w^{wk} and c_w^{wn} refer to the same meaning. To do so, we apply a simple maximum set overlap metric to their feature values. Let *F(c)* be a function that returns all feature values of *c* as bag-of-words, then for each $c_w^{wk} \in C_w^{wk}$, it is mapped to a c_w^{wn} such that $|F(c_w^{wn})| \cap |F(c_w^{wk})|$ is maximized among all $c_w^{wn} \in C_w^{wn}$. The resulting concept candidates are denoted as C_w^{mapd}, where $c_w^{mapd} = \{c_w^{wk}, c_w^{wn}\}$ is a mapped set of concepts potentially referring to the same meaning. If $C_w^{wn} = \emptyset$ then $C_w^{mapd} = C_w^{ref}, i.e., C_w^{wk}$.

Next, **cross-source feature enrichment** creates a uniform feature representation for each mapped sets of concepts. The process can be considered as enriching the features from one knowledge source with others. The most straightforward approach is to simply collect features extracted from each knowledge source on to a single graph, retaining the diversity in feature types. For example, Figure 2 shows a graph representation based on the collection of the four Wikipedia features and the four derived WordNet features. We refer to this approach as "***feature combination***".

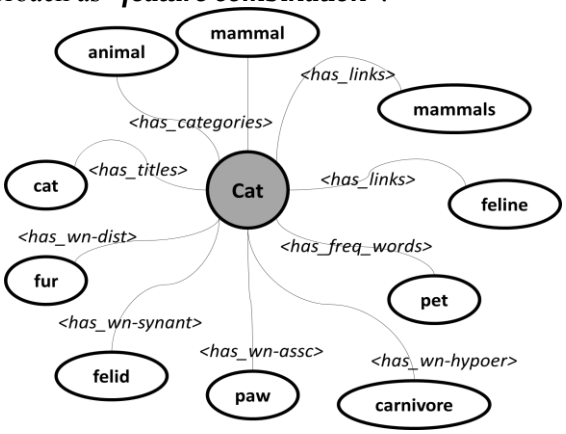


Figure 2. Representation of "cat, the mammal" after ***concept mapping*** and ***feature combination***

On the other hand, cross-source features may be merged according to their semantics. For example, WordNet and Wikipedia contain features based on synonyms of concepts; while Wikipedia and unstructured documents contain word distributional features. Thus we define "***feature integration***" as merging feature types from different knowledge sources into single types of features based on their similarity in semantics. With WordNet and Wikipedia, we integrate features as below (Figure 3):

- *merged-synant* merges Wikipedia page titles and redirection links with *wn-synant*
- *merged-hypoer* merges merges Wikipedia categories with *wn-hypoer*

- *merged-assc* merges Wikipedia links with *wn-assc*. We consider Wikipedia links bear other associative relations and are therefore merged with features extracted by other WordNet relations
- *merged-dist* merges Wikipedia frequent *n* words with *wn-dist*.

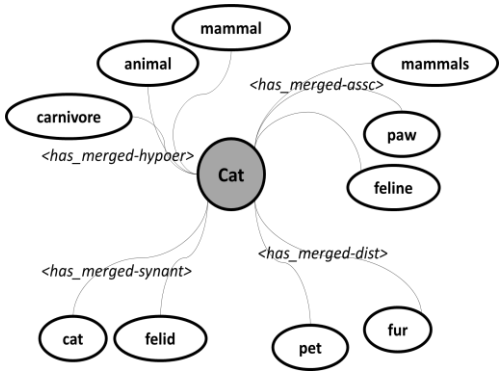


Figure 3. Representation of "cat, the mammal" after ***concept mapping*** and ***feature integration***

Note that the difference between cross-source ***feature combination*** and ***integration*** is that the former introduces more *types* of features, whereas the latter retains same number of feature types but increases *feature values* for each type. Both have the effect of establishing additional path (via features) between concepts, but in different ways.

With intra-resource **feature diversification,** cross-source ***feature combination*** and ***feature integration***, we create a total of nine intra- and cross-source feature representations to be tested with the uniform random walk model:

- four types of Wikipedia features (*wk-4F*)
- one type of Wikipedia features (*wk-1F*)
- ten types of WordNet features (*wn-10F*)
- four types of WordNet features (*wn-4F*)
- one type of WordNet features (*wn-1F*)
- *wk-4F combines wn-4F: wk-4F+wn4F,C*
- *wk-4F integrates wn-4F: wk-4F+wn4F,I*
- *wk-1F combines wn-1F: wk-1F+wn1F,C*
- *wk-1F integrates wn-1F: wk-1F+wn1F,I*

3.4 Computing SR using the graph

The algorithm for computing SR using the graph is based on the idea of random walk. It formalizes the idea that taking successive steps along the paths in a graph, the "easier" it is to arrive at a target node starting from a source node, the more related the two nodes are. Following the previous steps, the feature representations of all candidate concepts relevant to the input word pairs are joined, which creates a single undirected, weighted, bi-partite graph. Let *G = (V, E)* be the graph, where *V* is the set of nodes (concepts and feature values); *E* is the set of edges (feature types) that connect concepts and features. As shown in Figure 4, different concepts are connected if they share same values of same types of features, namely, there exists a path that connects one concept to another.

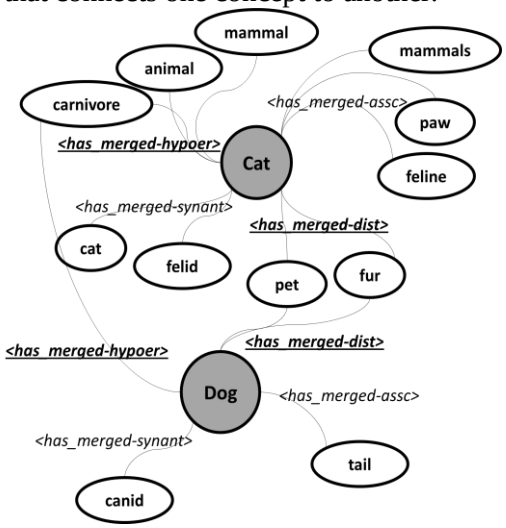


Figure 4. Paths are established between different concepts if they share values of same feature types <**bold underlined**>

Using Figure 4 it is easier to comprehend the difference between ***feature combination*** and ***integration***. Since concept nodes can only be connected by same types of edges (feature types), ***feature combination*** increases the chances of connectivity by adding in more types of edges, while ***integration*** merges similar types of edges across knowledge sources and increases the number of feature nodes connected by each type.

From the graph, we start by building an adjacency matrix *W* of initial probability distribution:

$$W_{ij} = \begin{cases} \sum_{l_k \in L} \frac{w(l_k)}{|(i,\cdot) \in E : l(i,\cdot) = l_k|}, (i,j) \in E \\ 0, otherwise \end{cases} \quad [1]$$

Where W_{ij} is the i^{th}-line and j^{th}-column entry of *W*, indexed by *V; l(i, j)* is a function that returns the type of edge (i.e., type of feature) connecting nodes *i* and *j*; *L* is the set of all possible types; *w(l)* returns the weight for that type. Essentially, *L* is the collection of all feature types, and *w(l)* assigns

a weight to a particular feature type. Next, we compute the transition probability matrix $P^{(t)}(j|i) = [(D^{-1}W)^t]_{ij}$ ($D_{ii} = \sum_k W_{ik}$), which returns the probability of reaching other nodes from a starting node on the graph after *t* steps. In this method, we follow the work by Rowe and Ciravegna (2010) to set *t=2* in order to preserve locally connected nodes. Next, we extract the probability vectors corresponding to concept nodes from *P*, and compute pair-wise relatedness using the cosine function. Effectively, this formalizes the notion that two concepts related to a third concept is also semantically related, which is similar to the hypothesis proposed by Patwardhan and Pedersen (2006) in their method based on second-order context vectors. The final SR between the input word pair is the maximum pair-wise concept SR.

4 Experiment and evaluation

We evaluate the method based on correlation against human judgment (gold standard) on seven benchmarking datasets covering both general and technical domains. These include four general domain datasets: the Rubenstein and Goodenough (1965) dataset containing 65 pairs of nouns (RG65); the Miller and Charles (1991) dataset that is a subset of the RG-65 dataset and contains 30 pairs (MC30); the Finkelstein et al. (2002) dataset with 353 pairs of words, including nouns, verbs, adjectives, as well as named entities. This contains two subsets, a set of 153 pairs (Fin153) and a set of 200 (Fin200) pairs each annotated by a different groups of annotators. Zesch and Gurevych (2010) show largely varying Inter-Annotator-Agreement (IAA) between the two sets (Table 1), and argue that they should be treated as separate datasets. Three biomedical datasets are selected to evaluate domain-specific performance of the proposed method. These include a set of 36 MeSH term pairs in Petrakis et al. (2006) (MeSH36), 30 pairs of medical terms annotated by a group of physicians as in Pedersen et al. (2006) (Ped30-p) and the same set annotated by a different group of medical coders (Ped30-c). Table 1 shows statistics of the seven datasets.

The correlation is computed using the Spearman rank order coefficient for two reasons. First, it is a better metric than other alternatives (Zesch and Gurevych, 2010). Second, it is consistent with the majority of studies such that results can be compared.

Dataset	Size	Domain	IAA
MC30	30	General	0.9
RG65	65	General	0.8
Fin153	153	General	0.73
Fin200	200	General	0.55
Ped30-p	30	Biomedical	0.68
Ped30-c	30	Biomedical	0.78
MeSH36	36	Biomedical	-

Table 1: Information of benchmarking datasets

We distribute feature weights *w(l)* across different feature types *L* evenly in each feature representation. Although Zhang et al. (2010) show that discriminated feature weights leads to improved accuracy; this is not the focus of this study. Since we aim to investigate the effects of harnessing different knowledge sources, we obtained baseline performances by applying the method to those feature representations based on single knowledge sources (i.e., *wk-4F, wk-1F, wn-10F, wn-4F, wn-1F*). Tables 2 and 3 show the best results obtained with baselines and corresponding knowledge sources and feature representation.

Dataset	Corr.	Feature	Coverage (% pairs)
MC30	0.77	*wn-1F*	77%
RG65	0.71	*wn-1F*	65%
Fin153	0.45	*wn-4F*	82%
Fin200	0.35	*wn-4F*	76%
Ped30-p	0.66	*wn-4F*	33%
Ped30-c	0.8	*wn-4F*	33%
MeSH36	0.49	*wn-1F*	50%

Table 2: Correlation obtained using *WordNet*. Many word pairs are not covered due to sparse feature space and lack of coverage. **Only covered pairs are accounted**.

Dataset	Corr.	Feature
MC30	0.74	*wk-1F*
RG65	0.67	*wk-1F*
Fin153	0.7	*wk-1F*
Fin200	0.51	*wk-4F*
Ped30-p	0.53	*wk-4F*
Ped30-c	0.58	*wk-4F*
MeSH36	0.73	*wk-4F*

Table 3: Correlation obtained using only Wikipedia. All word pairs are 100% covered.

Tables 4 – 6 show results obtained with enriched feature representation.

	Combination (**C**)		Integration (**I**)	
Dataset	*wn-4F + wk-4F*	*wn-1F + wk-1F*	*wn-4F + wk-4F*	*wn-1F + wk-1F*
MC30	0.77	0.8	0.8	0.79
RG65	0.74	0.73	0.73	0.729
Fin153	0.73	0.75	0.74	0.73
Fin200	0.52	0.54	0.53	0.54
Ped30-p	0.63	0.52	0.64	0.47
Ped30-c	0.64	0.52	0.67	0.49
MeSH36	0.7	0.694	0.75	0.7

Table 4: Correlation obtained using both knowledge sources. Word pairs are 100% covered.

	KS and # of feature types			
	WN	WK	WK+WN,**C**	WK+WN, **I**
MC30	1	1	1	4
RG65	1	1	4	4
Fin153	4	1	1	4
Fin200	4	4	1	1
Ped30-p	4	4	4	4
Ped30-c	4	4	4	4
MeSH36	1	4	4	4

Table 5: Number of feature types with which best results are obtained on each dataset. **KS**: Knowledge Source

	Single KS	Multiple KS		Impr.
Dataset	Best corr.	Best corr.	Strategy	
MC30	0.74	0.8	C/I	0.06
RG65	0.67	0.74	C	0.07
Fin153	0.7	0.75	C	0.05
Fin200	0.51	0.54	C/I	0.03
Ped30-p	0.53	0.64	I	0.11
Ped30-c	0.58	0.67	I	0.09
MeSH36	0.73	0.75	I	0.02

Table 6: Improvement achieved by harnessing multiple KSs. Best correlation with single KS is based on Wikipedia, which provides 100% coverage of word pairs.

Tables 7 and 8 compare our method against state-of-the-art. For Table 8, figures for other state-of-the-art systems can be found in corresponding publications; while we only list the best performing systems for comparison.

	MC30	RG65	Fin153	Fin200	KS
best of WN+WK	**0.8**	0.74	**0.75**	**0.54**	Both
Rad89*	0.75	0.79	0.33	0.24	WN
LC98*	0.75	0.79	0.33	0.24	WN
WP94*	0.77	0.78	0.38	0.24	WN
HS98*	0.76	0.79	0.33	0.32	WN
Res95*	0.72	0.74	0.35	0.26	WN
JC97*	0.68	0.58	0.28	0.10	WN
Lin98*	0.67	0.60	0.27	0.17	WN
Zes07*	0.77	0.**82**	0.6	0.51	WK
GM07*	0.67	0.75	0.69	0.51	WK
Zha10	0.71	0.76	0.71	0.46	WK

Table 7[3]: Comparison against state-of-the-art in the general domain. (* figures from Zesch and Gurevych, 2010)

	Ped30-p	Ped30-c	MeSH36	KS
best of WN+WK	0.64	0.67	**0.75**	WN+ WK
Pet06 best	-	-	0.74	MeSH
Ped06 best	**0.84**	**0.75**	-	GO, D
Ped06 second	0.62	0.68	-	GO, D

Table 8[4]: Comparison against state-of-the-art in the biomedical domain. GO – Gene Ontology; D – document sets.

Given the fact that some datasets (i.e., MC30, Ped30-p, Ped30-c, MeSH36) have a relatively low sample size, we cannot always be sure that correlation values are accurate or occurred by chance. Therefore, we measure the statistical significance of correlation by computing the *p-value* for the correlation values reported for our system in Tables 7 and 8. For all cases, a *p-value* of less than 0.001 is obtained, which indicates that correlation values are statistically significant.

[3] Rada (1989) (Rad89); Leacock and Chodorow (1998) (LC98); Wu and Palmer (1994) (WP04); Hirst and St-Onge (1998) (HS98); Resnik (1995) (Res95); Jiang and Conrath (1997) (JC97); Lin (1998) (Lin98); Zesch and Gurevych (2007) (ZG07); Gabrilovich and Markovitch (2007) (GM07); Zhang et al. (2010) (Zha10)

[4] Petrakis et al. (2006) (Pet06); Pedersen et al. (2006) (Ped06). Original participating systems can be found in these works.

5 Discussion

Single v.s. multiple knowledge sources As shown in Table 6, considering the best performances across all feature enrichment strategies and feature sets, the proposed method successfully harnessed different knowledge sources and improved over the baselines using single knowledge sources by 0.02 ~ 0.11. The biggest improvement (0.11) is on a domain-specific dataset, on which the method based on single knowledge source performed poorly in terms of coverage and accuracy. The best enrichment strategy that has consistently improved the baselines is *wk-4F+wn-4F, Integration* (Table 4 v.s. Table 3). With features enriched from *multiple* knowledge sources, the method also consistently improved over their *corresponding single*-source features on all datasets, except MeSH36, on which *wk-4F+wn-4F, Combination* (Table 4) slightly reduced the accuracy obtained with *wk-4F* (Table 3) only.

The large proportion of uncovered word pairs using WordNet is due to its lack of coverage of specialized lexicons, and sparser semantic content. For example, of all 115 distinctive terms in the Ped30 and MeSH36 datasets, 30% are not included in WordNet. And of all 447 distinctive words in all general domain datasets, only 69% have multiple synonyms. Features such as *attributes* and "*see also*" are present for less than 20 words. This is the reason that some approaches using WordNet (e.g., Agirre et al., 2009) require a graph of all WordNet lexicons to be built, thus intermediate words may "bridge" input words even if they do not connect directly by their features. Nevertheless, the improvement in accuracy and 100% coverage after harnessing both knowledge sources suggests that they complement each other well. On one hand, Wikipedia brings its strength in domain and content coverage; on the other hand, WordNet brings useful semantic evidences for words that are covered.

Concept mapping and feature enrichment methods While the set overlap based method for cross-source concept mapping using the reference knowledge source concepts is simple and proved successful, the accuracy of mapping and its correlation with the accuracy of the SR method was not studied. This will be explored in the future. Also, alternative mapping methods will be investigated. For example, Toral and Muñoz (2006) describe a different method of mapping Wikipedia articles to WordNet synsets; one could also adopt a simple disambiguation process to select the best candidate concept from each knowledge source suited for the input word pairs, whereby cross-source concept mapping becomes straightforward. In terms of feature enrichment strategies, there is no strong indication (Table 6) of which (*feature combination v.s. integration*) is more effective, although the system consistently outperforms the baselines (Table 4 v.s. Table 3) with the *wk-4F+wn-4F, Integration* strategy.

Feature diversification v.s. unification Table 5 suggests that in most cases, differentiating feature types leads to better results than merging them uniformly, despite the knowledge sources used. This is consistent with the findings by Zhang et al. (2010). This can be understandable since although unifying feature types effectively increases possibility of sharing features, equally, this may also increase the proportion of noisy features. For example, consider the Wikipedia article of "Horse" (animal), which has a category label "livestock"; and the article "Famine", which has an outgoing link "livestock" (in a sentence describing diseases that caused decline of livestock production). By differentiating the feature types "*has_category*" and "*has_outlink*", the two concepts will not be connected even if they both have the same word "livestock" in their feature representation. However, using a bag-of-words representation where feature types are undistinguished, the strength of their relatedness is boosted by sharing this word, which may be uninteresting in this occasion.

Compared against state-of-the-art, the proposed method has achieved promising results. Overall, by harnessing different knowledge sources, the method achieves, and in many cases, outperforms state-of-the-art. In the general domain, it outperforms state-of-the-art on three out of four datasets. It is worth noting that all methods based on WordNet generally have poor performance on the Fin153 and Fin200 datasets (Table 7). Despite the heterogeneity in these datasets, this may also relate to the quality of the feature space generated with WordNet. In fact methods using Wikipedia perform better on these datasets. With enriched features from both knowledge sources, the accuracies are further improved.

In the biomedical domain, the proposed method outperforms state-of-the-art on one dataset and produces competitive results on others. Note that all other methods exploit domain-specific ontologies and corpora. The *Ped06 best* and *Ped06 second* methods also depend on a corpus of one million documents. These results further confirmed the benefits of our method: harnessing knowledge from general-purpose knowledge sources of limited domain coverage, it is possible to achieve results that rival methods based on well-curated and specially tailored domain-specific knowledge sources. This is an encouraging finding. Although there are abundant resources in the biomedical domain for this type of tasks, such resources may be scarce in other domains and are expensive to build. However, the results suggest that the proposed method offers a more affordable approach that provides reasonable coverage and quality, even if individual general knowledge sources may be limited in themselves.

Generality of the method The proposed method represents features extracted from different knowledge sources in a generic manner, which facilitates cross-source feature enrichment and requires generic algorithm computation. As discussed in Section 3, semantic evidence of words and concepts may be extracted from different knowledge sources in different ways, while harnessed in the generic model. In contrast, other methods using multiple knowledge sources (e.g., Han and Zhao, 2010; Tsang and Stevenson, 2010) introduce algorithms that are bound to the knowledge sources, which may limit their adaptability and portability.

6 Conclusion

This paper introduced a generic method of harnessing different knowledge sources to compute semantic relatedness. We have shown empirically that different knowledge sources contain complementary semantic evidence, which, when combined together under a uniform model, can improve the accuracy of SR methods. Moreover, we have demonstrated its robustness in dealing with knowledge sources of different quality and coverage. Several remaining issues will be studied in the future. First, additional knowledge sources will be studied, particularly unstructured corpora and domain-specific resources. The experiments have shown that although harnessing different knowledge sources achieved encouraging results on biomedical datasets, they are still far from being perfect. While it should be appreciated that the results are obtained using only general purpose knowledge sources, it would be interesting to investigate whether harnessing domain specific knowledge sources (where available) further improves the performance. Second, different methods of concept mapping will be studied. We will also design methods for assessing the quality of mapping, and analyze their correlations with the SR methods. Third, analyses will be carried out to uncover the differences between feature combination and integration that have led to different accuracies.

Acknowledgments

Part of this research has been funded under the EC 7th Framework Program, in the context of the SmartProducts project (231204).

References

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., Soroa, A. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In Proceedings of NAACL'09

Batet, M., Sánchez, D., Valls, A. 2010. An ontology-based measure to compute semantic similarity in biomedicine. In Journal of Biomedical Informatics, **44**(1), 118-125

Chen, H., Lin, M., Wei, Y. 2006. Novel association measures using web search with double checking. Proceedings of COLING'06-ACL'06, pp. 1009-1016

Cilibrasi, R., Vitanyi, P. 2007. The Google Similarity Distance. In IEEE Transactions on Knowledge and Data Engineering. 19(3), 370-383

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: the concept revisited. In ACM Transactions on Information Systems, 20 (1), pp. 116 – 131

Gabrilovich, E., Markovitch, S. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In proceeding of IJCAI'07

Gouws, S., Rooyen, G., Engelbrecht, H. 2010. Measuring conceptual similarity by spreading activation over Wikipedia's hyperlink structure. Proceedings of the 2nd Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources

Halavais , A. 2008. An Analysis of Topical Coverage of Wikipedia. Journal of Computer-Mediated Communication, **13**(2)

Han, X., Zhao, J. 2010. Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In the 48th Annual Meeting of the Association for Computational Linguistics.

Harrington, B. 2010. A semantic network approach to measuring relatedness. In Proceedings of COLING' 10

Hirst, G., and St-Onge, D. 1998. Lexical chains as representation of context for the detection and correction malapropisms. In Christiane Fellbaum (ed.), WordNet: An Electronic Lexical Database and Some of Its Applications, pp. 305–332. Cambridge, MA: The MIT Press.

Holloway, T., Bozicevic, M., Börner, K. 2007. Analyzing and visualizing the semantic coverage of Wikipedia and its authors. In Journal of Complexity, Special issue on Understanding Complex Systems, **12**(3), 30-40

Jiang, J. and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. Proceedings of the International Conference on Research in Computational Linguistics, pp. 19-33

Leacock, C., Chodorow, M. 1998. Combining local context and WordNet similarity for word sense identification. In C. Fellbaum (Ed.), WordNet. An Electronic Lexical Database, Chp. 11, pp. 265-283.

Lin, D. 1998. An information-theoretic definition of similarity. Proceedings of the Fifteenth International Conference on Machine Learning, pp. 296-304

Lord, P., Stevens, R., Brass, A., Goble, C. 2003. Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. In Bioinformatics, **19**(10), pp. 1275–1283

Matsuo, Y., T. Sakaki., K., Uchiyama, M., Ishizuka. 2006. Graph-based word clustering using a web search engine. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp.542-550

Miller, G., Charles, W. 1991. Contextual correlates of semantic similarity. In Language and Cognitive Processes, 6(1): 1-28

Pan, F., Farrell, R. 2007. Computing semantic similarity between skill statements for approximate matching. In Proceedings of NAACL-HLT'07, pp. 572-579

Patwardhan, S., Pedersen, T. 2006. Using WordNet-based context vectors to estimate the semantic relatedness of concepts. Proceedings of the EACL 2006 Workshop on Making Sense of Sense: Bringing Computational Linguistics and Psycholinguistics Together

Pedersen, T., Pakhomov, S., Patwardhan, S., Chute, C. 2006. Measures of semantic similarity and relatedness in the biomedical domain. Journal of Biomedical Informatics **40**(3), 288-299

Pekar, V., Staab, S. 2002. Taxonomy learning: factoring the structure of a taxonomy into a semantic classification decision. Proceedings of COLING'02. pp. 786-792

Pesquita, C., Faria, D., Bastos, H., Falcão, A., Couto, F. (2007). Evaluating GO-based Semantic Similarity Measures. ISMB/ECCB 2007 SIG Meeting Program Materials, International Society for Computational Biology 2007

Petrakis, E., Varelas, G., Hliaoutakis, A., Raftopoulou, P. 2006. Design and evaluation of semantic similarity measures for concepts stemming from the same or different ontologies. In 4th Workshop on Multimedia Semantics (WMS'06), pp. 44-52.

Pirro, G. 2009. A semantic similarity metric combining features and intrinsic information content. In Data and Knowledge Engineering, **68**(11), pp. 1289-1308

Pozo A., Pazos F., Valencia, A. 2008. Defining functional distances over gene ontology. In BMC Bioinformatics **9**, pp.50

Rada, R., Mili, H., Bicknell, E., Blettner, M. 1989. Development and application of a metric on semantic nets. In IEEE Transactions on Systems, Man and Cybernetics **19**(1), pp.17-30

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of IJCAI-95, pp. 448-453

Riensche, R., Baddeley, B., Sanfilippo, A., Posse, C., Gopalan, B. 2007. XOA: Web-Enabled Cross-Ontological Analytics. IEEE Congress on Services, pp. 99-105

Rowe, M., Ciravegna, F. 2010. Disambiguating identity web references using Web 2.0 data and semantics. M Rowe and F Ciravegna. The Journal of Web Semantics.

Rubenstein, H., Goodenough, J. 1965. Contextual correlates of synonymy. In Communications of the ACM, 8(10):627-633

Seco, N., and Hayes, T. 2004. An intrinsic information content metric for semantic similarity in WordNet. In Proceedings of the 16th European conference on Artificial Intelligence

Strube, M., Ponzetto, S. 2006. WikiRelate! Computing semantic relatedness using Wikipedia. In Proceedings of the 21st national conference on Artificial intelligence (AAAI)

Toral, A., Muñoz, R. 2006. A Proposal to Automatically Build and Maintain Gazetteers for Named Entity Recognition by using Wikipedia. In Proceedings of Workshop on New Text, ACL'06.

Tsang, V., Stevenson, S. 2010. A graph-theoretic framework for semantic distance. In Journal of Computational Linguistics, 36(1).

Wojtinnek, P., Pulman, S. 2011. Semantic relatedness from automatically generated semantic networks. In Proceedings of the Ninth International Conference on Computational Semantics (IWCS'11)

Wu, Z. Palmer, M. 1994. Verbs semantics and lexical selection. Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pp. 133-138

Wu, H., Su, Z., Mao, F., Olman, V., Xu, Y. 2005. Prediction of functional modules based on comparative genome analysis and gene ontology application. Nucleic Acids Research, **33**, pp. 2822–2837.

Yazdani, M., Popescu-Belis, A. 2010. A random walk framework to compute textual semantic similarity: a unified model for three benchmark tasks. IEEE Fourth International Conference on Semantic Computing (ICSC), pp. 424-429

Ye, P., Peyser, B., Pan, X., Boek, J., Spencer, F., Bader, J. 2005. Gene function prediction from congruent synthetic lethal interactions in yeast. In Molecular system biology

Yeh, E., Ramage, D., Manning, C., Agirre, E., Soroa, A. 2009. WikiWalk: random walks on Wikipedia for semantic relatedness. In Proceedings of the TextGraphs-4, Workshop on Graph-based Methods for Natural Language Processing, ACL2009

Zesch, T., and Gurevych, I. 2007. Analysis of the Wikipedia category graph for NLP applications. In Proceedings of the TextGraphs-2 Workshop (NAACL-HLT 2007), pp. 1–8

Zesch, T., Gurevych, I. 2010. Wisdom of crowds versus wisdom of linguists: measuring the semantic relatedness of words. In Journal of Natural Language Engineering, **16**, pp. 25-59

Zhang, Z., Gentile, A., Xia, L., Iria, J., Chapman, S. 2010. A random graph walk based approach to compute semantic relatedness using knowledge from Wikipedia. In Proceedings of LREC'10.

Refining the Notions of Depth and Density in WordNet-based Semantic Similarity Measures

Tong Wang
Department of Computer Science
University of Toronto
tong@cs.toronto.edu

Graeme Hirst
Department of Computer Science
University of Toronto
gh@cs.toronto.edu

Abstract

We re-investigate the rationale for and the effectiveness of adopting the notions of depth and density in WordNet-based semantic similarity measures. We show that the intuition for including these notions in WordNet-based similarity measures does not always stand up to empirical examination. In particular, the traditional definitions of depth and density as ordinal integer values in the hierarchical structure of WordNet does not always correlate with human judgment of lexical semantic similarity, which imposes strong limitations on their contribution to an accurate similarity measure. We thus propose several novel definitions of depth and density, which yield significant improvement in degree of correlation with similarity. When used in WordNet-based semantic similarity measures, the new definitions consistently improve performance on a task of correlating with human judgment.

1 Introduction

Semantic similarity measures are widely used in natural language processing for measuring distance between meanings of words. There are currently two mainstream approaches to deriving such measures, i.e., distributional and lexical resource-based approaches. The former usually explores the co-occurrence patterns of words in large collections of texts such as text corpora (Lin, 1998) or the Web (Turney, 2001). The latter takes advantage of mostly handcrafted information, such as dictionaries (Chodorow et al., 1985; Kozima and Ito, 1997) or thesauri (Jarmasz and Szpakowicz, 2003).

Another important resource in the latter stream is semantic taxonomies such as WordNet (Fellbaum, 1998). Despite their high cost of compilation and limited availability across languages, semantic taxonomies have been widely used in similarity measures, and one of the main reasons behind this is that the often complex notion of lexical semantic similarity can be approximated with ease by the distance between words (represented as nodes) in their hierarchical structures, and this approximation appeals much to our intuition. Even methods as simple as "hop counts" between nodes (e.g., that of Rada et al. 1989 on the English WordNet) can take us a long way. Meanwhile, taxonomy-based methods have been constantly refined by incorporating various structural features such as depth (Sussna, 1993; Wu and Palmer, 1994), density (Sussna, 1993), type of connection (Hirst and St-Onge, 1998; Sussna, 1993), word class (sense) frequency estimates (Resnik, 1999), or a combination these features (Jiang and Conrath, 1997). Most of these algorithms are fairly self-contained and easy to implement, with off-the-shelf toolkits such as that of Pedersen et al. (2004).

With the existing literature focusing on carefully weighting these features to construct a better semantic similarity measure, however, the rationale for adopting these features in calculating semantic similarity remains largely intuitive. To the best of our knowledge, there is no empirical study directly investigating the effectiveness of adopting structural features such as depth and density. This serves as the major motivation for this study.

The paper is organized as follows. In Section 2 we review the basic rationale for adopting depth

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1003–1011,
Edinburgh, Scotland, UK, July 27–31, 2011.

and density in WordNet-based similarity measures as well as existing literature on constructing such measures. In Section 3, we show the limitations of the current definitions of depth and density as well as possible explanations for these limitations.[1] We then propose new definitions to avoid such limitations in Section 4. The effectiveness of the new definitions is evaluated by applying them in semantic similarity measures in Section 5 and conclusions made in Section 6.

2 Related Work

The following are the current definitions of depth and density which we aim at improving. Given a node/concept c in WordNet, depth refers to the number of nodes between c and the root of WordNet, (i.e., the root has depth zero, its hyponyms depth one, and so on). There are more variations in the definition of density, but it is usually defined as the number of edges leaving c (i.e., its number of child nodes) or leaving its parent node(s) (i.e., its number of sibling nodes). We choose to use the latter since it is used by most of the existing literature.

2.1 The Rationale for Depth and Density

The rationale for using the notions of depth and density in WordNet-based semantic similarity measures is based on the following assumption:

Assumption 1 *Everything else being equal, two nodes are semantically closer if (a) they reside deeper in the* WordNet *hierarchy, or (b) they are more densely connected locally.*

This is the working assumption for virtually all WordNet-based semantic similarity studies using depth and/or density. For depth, the intuition is that adjacent nodes deep down the hierarchy are likely to be conceptually close, since the differentiation is based on finer details (Jiang and Conrath, 1997). Sussna (1993) termed the use of depth as *depth-relative scaling*, claiming that "only-siblings deep in a tree are more closely related than only-siblings higher in the tree". Richardson and Smeaton (1995) gave an hypothetical example illustrating this "only-siblings" situation, where *plant–animal* are the only two nodes under *living things*, and *wolfhound–foxhound* under *hound*. They claimed the reason that the former pair can be regarded as conceptually farther apart compared to the latter is related to the difference in depth.

As for the relation between density and similarity, the intuition is that if the overall semantic mass for a given node is constant (Jiang and Conrath, 1997), then the more neighboring nodes there are in a locally connected subnetwork, the closer its members are to each other. For example, *animal*, *person*, and *plant* are more strongly connected with *life form* than *aerobe* and *plankton* because the first three words all have high density in their local network structures (Richardson and Smeaton, 1995). Note that the notion of density here is not to be confused with the *conceptual density* used by Agirre and Rigau (1996), which is essentially a semantic similarity measure by itself.

In general, both observations on depth and density conform to intuition and are supported qualitatively by several existing studies. The main objective of this study is to empirically examine the validity of this assumption.

2.2 Semantic Similarity Measures Using Depth and/or Density

One of the first examples of using depth and density in WordNet-based similarity measures is that of Sussna (1993). The weight on an edge between two nodes c_1 and c_2 with relation r in WordNet is given as:

$$w(c_1, c_2) = \frac{w(c_1 \rightarrow_r c_2) + w(c_2 \rightarrow_r c_1)}{2d}$$

where d is the depth of the deeper of the two nodes. As depth increases, weight decreases and similarity in turn increases, conforming to Assumption 1. The edge weight was further defined as

$$w(c_1 \rightarrow_r c_2) = max_r - \frac{max_r - min_r}{n_r(c_1)}$$

where $n_r(X)$ is "the number of relations of type r leaving node X", which is essentially an implicit form of density, and max_r and min_r are the maximum and minimum of n_r, respectively. Note that this formulation of density contradicts Assumption

[1] Since the works we review in this section have different definitions of depth and density, we defer our formal definitions to Section 3.

1 since it is proportional to edge weight (left-hand-side) and thus negatively correlated to similarity.

Wu and Palmer (1994) proposed a concept similarity measure between two concepts c_1 and c_2 as:

$$sim(c_1,c_2) = \frac{2 \cdot dep(c)}{len(c_1,c) + len(c_2,c) + 2 \cdot dep(c)} \quad (1)$$

where c is the lowest common subsumer (LCS) of c_1 and c_2, and $len(\cdot,\cdot)$ is the number of edges between two nodes. The rationale is to adjust "hop count" (the first two terms in the denominator) with the depth of LCS: similarity between nodes with same-level LCS is in negative proportion to hop counts, while given the same hop count, a "deeper" LCS pulls the similarity score closer to 1.

Jiang and Conrath (1997) proposed a hybrid method incorporating depth and density information into an information-content-based model (Resnik, 1999):

$$\begin{aligned} w(c,p) =& (\frac{dep(p)+1}{dep(p)})^{\alpha} \\ & \times [\beta + (1-\beta)\frac{\bar{E}}{den(p)}] \\ & \times [IC(c) - IC(p)]T(c,p) \end{aligned} \quad (2)$$

Here, p and c are parent and child nodes in WordNet, $dep(\cdot)$ and $den(\cdot)$ denote the depth and density of a node, respectively, $\bar{E}$ is the average density over the entire network of WordNet, and α and β are two parameters controlling the contribution of depth and density values to the similarity score. $IC(\cdot)$ is the information content of a node based on probability estimates of word classes from a small sense-tagged corpus (Resnik, 1999), and $T(c,p)$ is a link-type factor differentiating different types of relations between c and p.

3 Limitations on the Current Definitions of Depth and Density

To what extent do the notions of depth and density help towards an accurate semantic similarity measure? Our empirical investigation below suggests that more often than not, they fail our intuition.

A direct assessment of the effectiveness of using depth and density is to examine their correlation with similarity. Empirical results in this section

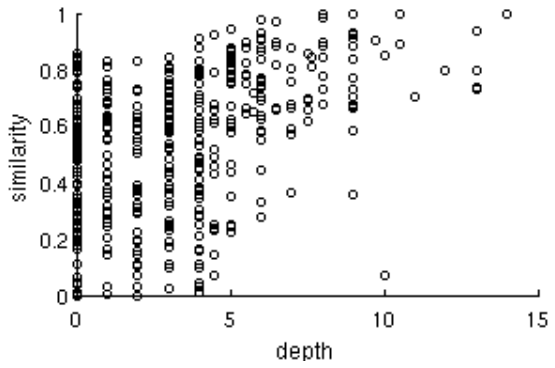


Figure 1: Correlation between depth and similarity.

are achieved by the following experimental setting. Depth is defined as the number of edges between the root of the hierarchy and the lowest common subsumer (LCS) of two nodes under comparison, and density as the number of siblings of the LCS.[2] Similarity is measured by human judgment on similarity between word pairs. Commonly used data sets for such judgments include that of Rubenstein and Goodenough (1965), Miller and Charles (1991), and Finkelstein et al. (2001) (denoted *RG*, *MC*, and *FG*, respectively). *RG* is a collection of similarity ratings of 65 word pairs averaged over judgments from 51 human subjects on a scale of 0 to 4 (from least to most similar). *MC* is a subset of 30 pairs out of the *RG* data set. These pairs were chosen to have evenly distributed similarity ratings in the original data set, and similarity judgment was elicited from 38 human judges with the same instruction as used for *RG*. *FG* is a much larger set consisting of 353 word pairs, and the rating scale is from 0 to 10. We combine the *RG* and *FG* data sets in order to maximize data size. Human ratings r on individual sets are normalized to r_n on 0 to 1 scale by the following formula:

$$r_n = \frac{r - r_{\min}}{r_{\max} - r_{\min}}$$

where $r_{\max}$ and $r_{\min}$ are the maximum and minimum of the original ratings, respectively. Correlation is evaluated using *Spearman's* ρ.

[2] We also tried several other variants of these definitions, e.g., using the maximum or minimum depth of the two nodes instead of the LCS. With respect to statistical significance tests, these variants all gave the same results as our primary definition.

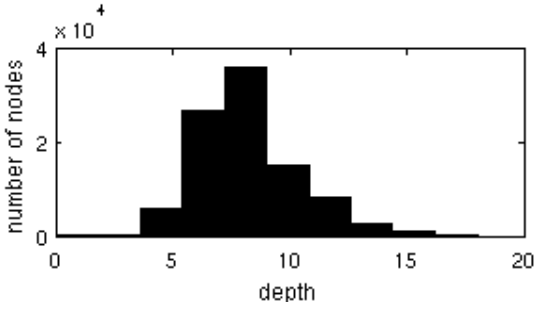


Figure 2: Histogram of depth of WordNet noun synsets.

3.1 Depth

The distribution of similarity of the combined data set over depth is plotted in Figure 1. For depth values under 5, similarity scores are fairly evenly distributed over depth, showing no statistical significance in correlation. For depth 5 and above, the shape of distribution resembles an upper-triangle, suggesting that (1) correlation with similarity becomes stronger in this range of depth value, and (2) data points with higher depth values tend to have higher similarity scores, but the reverse of the claim does not hold, i.e., word pairs with "shallower" LCS can also be judged quite similar by humans.

There are many more data points with lower depth values than with higher depth values in the combined data set. In order to have a fair comparison of statistical significance tests on the two value ranges for depth, we randomly sample an equal number (100) of data points from each value range, and the correlation coefficient between depth and similarity is averaged over 100 of such samplings. Correlation coefficients for depth value under 5 versus 5 and above are $\rho = 0.0881, p \approx 0.1$ and $\rho = 0.3779, p < 0.0001$, respectively, showing an apparent difference in degree of correlation.

Two interesting observations can be made from these results. Firstly, the notion of depth is relative to the distribution of number of nodes over depth value. For example, depth 20 by itself is virtually meaningless since it might be quite high if the majority of nodes in WordNet are of depth 10 or less, or quite low if the majority depth value are 50 or more. According to the histogram of depth values in WordNet (Figure 2), the distribution of number of nodes over depth value approximately conforms to a normal distribution $\mathcal{N}(8,2)$. It is visually quite noticeable that the actual quantity denoting how deep a node resides in WordNet is conflated at depth values below 5 or above 14. In other words, the distribution makes it rather inaccurate to say, for instance, that a node of depth 4 is twice as deep as a node of depth 2. This might explain the low degree of correlation between similarity and depth under 5 in Figure 1 (manifested by the long, vertical stripes across the entire range of similarity scores (0 to 1) for depth 4 and under), and also how the correlation increases with depth value. Unfortunately, we do not have enough data for depth above 14 to draw any conclusion on this higher end of the depth spectrum.

Secondly, even on the range of depth values with higher correlation with similarity, there is no definitive sufficient and necessary relation between depth and similarity (hence the upper triangle instead of a sloped line or band). Particularly, semantically more similar words are not necessarily deeper in the WordNet hierarchy. Data analysis reveals that the LCS of highly similar words can be quite close to the hierarchical root. Examples include *coast–shore*, which is judged to be very similar by humans (9 on a scale of 0–10 in both data sets). The latter is a hypernym of the former and thus the LCS of the pair, yet it is only four levels below the root node *entity* (via *geological formation*, *object*, and *physical entity*). Another situation is when the human judges confused relatedness with similarity, and WordNet fails to capture the relatedness with its hierarchical structure of lexical semantics: the pair *software–computer* can only be related by the root node *entity* as their LCS, although the pair is judged quite "similar" by humans (8.5 on 0 to 10 scale).

The only conclusive claim that can be made here is that word pairs with deeper LCS's tend to be more similar. However, since only word forms (rather than senses) are available in these psycho-linguistic experiments, the one similarity rating given by human judges sometimes fails to cover multiple senses for polysemous words. In the pair *stock–jaguar* of the *FG* set, for example, one sense of *stock* (*livestock, stock, farm animal: any animals kept for use or profit*) is closely connected to *jaguar* through a depth-10 LCS (*placental, placental mammal, eutherian, eutherian mammal*). However, the pair received a low similarity rating (0.92 on 0–10), prob-

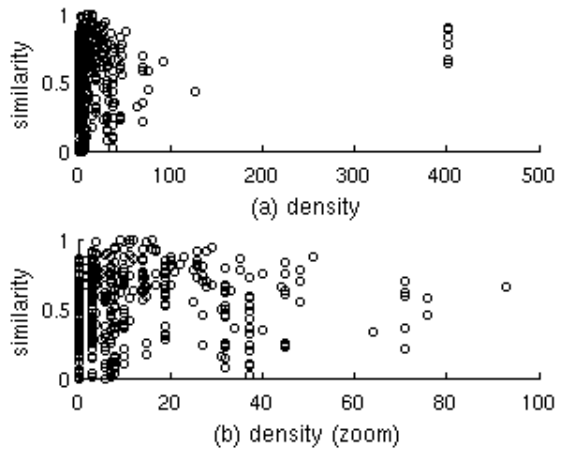


Figure 3: Correlation between density and similarity.

	MC	*RG*	*FG*
dep	0.7056***	0.6909***	0.3701***
den	0.2268	0.2660*	0.1023

Table 1: Correlation between depth/density and similarity on individual data sets. Number of asterisks indicates different confidence intervals ("*" for $p < 0.05$, "***" for $p < 0.0001$).

ably because judges associated the word form *stock* with its financial sense, especially when there was an abundant presence of pairs indicating this particular sense of the word (e.g., *stock–market*, *company–stock*).

3.2 Density

Comparing to depth, density exhibits much lower correlation with similarity (Figure 3-a and 3-b). We conducted correlation experiments between density and similarity with the same setting as for depth and similarity above. Data points with extremely high density values (up to over 400) are mostly idiosyncratic to the densely connected regions in WordNet and are numerically quite harmful. We thus excluded outliers with density values above 100 in the experiment.

Evaluation on the combined data set shows no correlation between density and similarity. To confirm the result, we break the experiments down to the three individual data sets, and the results are listed in Table 1. The correlation coefficient between density and similarity ranges from 0.10 to 0.27 There is no statistical significance of correlation on two of the three data sets (*MC* and *FG*), and the significance on *RG* is close to marginal with $p = 0.0366$.

Data analysis suggests that density values are often biased by particular fine-grainedness of local structures in WordNet. Qualitatively, Richardson and Smeaton (1995) previously observed that "the irregular densities of links between concepts results in unexpected conceptual distance measures". Empirically, on the one hand, more than 90% of WordNet nodes have density values less than or equal to 3. This means that for 90% of the LCS's, there are only three integer values for density to distinguish the varying degrees of similarity. In other words, such a range might be too narrow to have any real distinguishing power over similarity. On the other hand, there are outliers with extreme density values particular to the perhaps overly fine-grained subcategorization of some WordNet concepts, and these nodes can be LCS's of word pairs of drastically different similarity. The node *person*, *individual*, for example, can be the LCS of similar pairs such as *man–woman*, as well as quite dissimilar ones such as *boy–sage*, where the large density value does not necessarily indicate high degree of similarity.

Another crucial limitation of the definition of density is the information loss on specificity. In the existing literature, density is often adopted as a proxy for the degree of specificity of a concept, i.e., nodes in densely connected regions in WordNet are taken to be more specific and thus closer to each other. This information of a given node should be inherited by its hierarchical descendants, since specificity should monotonically increase as one descends the hierarchy. For example, the node *piano* has a density value of 15 under the node *percussion instrument*. However, the density value of its hyponyms *Grand piano*, *upright piano*, and *mechanical piano*, is only 3. Due to the particular structure of this subnetwork in WordNet, the *grand–upright* pair might be incorrectly regarded as less specific (and thus less similar) than, say, between *piano–gong*, both as percussion instruments.

4 New Definitions of Depth and Density

In this section, we formalize new definitions of depth and density to correct for their current limi-

	MC	*RG*	*FG*
dep_u	0.7201***	0.6798***	0.3751***
den_u	0.2268	0.2660*	0.1019
den_i	0.7338***	0.6751***	0.3445***

Table 2: Correlation between new definitions of depth/density and similarity.

tations discussed in Section 3.

4.1 Depth

The major problem with the current definition of depth is its failure to take into account the uneven distribution of number of nodes over the depth value. As seen in previous examples, the distribution is rather "flat" on both ends of depth value, which does not preserve the linearity of using the ordinal values of depth and thus introduces much inaccuracy.

To avoid this problem, we "re-curve" depth value to the cumulative distribution. Specifically, if we take the histogram distribution of depth value in Figure 2 as a probability density function, our approach is to project cardinal depth values onto its cumulative distribution function. The new depth is denoted dep_u and is defined as:

$$dep_u(c) = \frac{\sum_{c' \in WN} |\{c' : dep(c') \leq dep(c)\}|}{|WN|}$$

Here, $dep(\cdot)$ is the original depth value, and *WN* is the set of all nodes in WordNet. The resulting depth values not only reflect the flat ends, but also preserve linearity for the depth value range in the middle. In comparison with Table 1), correlation between dep_u and similarity increases over the original depth values on two of the three data sets (first row in Table 2 and decreases on the *RG* set. Later, in Section 5, we show how these marginal improvements translate into better similarity measures with statistical significance.

4.2 Density

In theory, a procedure analogous to the above cumulative definition can also be applied to density, i.e., by projecting the original values onto the cumulative distribution function. However, due to the Zipfian nature of density's histogram distribution (Figure 4, in contrast to Gaussian for depth in Figure 2), this is essentially to collapse most density values into a very small number of discrete values (which correspond to the original density of 1 to 3). Experiments show that it does not help in improving correlation with similarity scores (second row in Table 2 for den_u): correlation remains the same on *MC* and *RG*, and decreases slightly on *FG*.

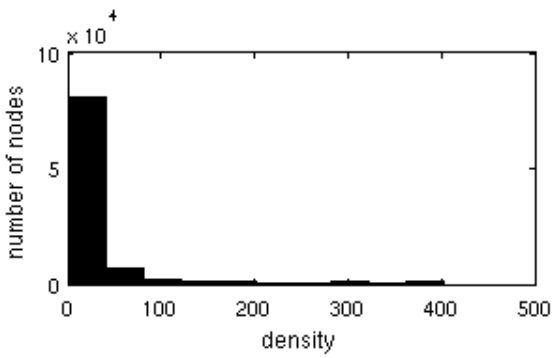


Figure 4: Histogram of density in WordNet.

We therefore resort to addressing the issue of information loss on specificity by inheritance. Intuitively, the idea is to ensure that a node be assigned no less density mass than its parent node(s). In the "piano" example (Section 3.2), the concept *piano* is highly specific due to its large number of siblings under the parent node *percussion instruments*. Consequently, the density of its child nodes *upright piano* and *grand piano* should inherit its specificity on top of their own.

Formally, we redefine density recursively as follows:

$$den_i(r) = 0$$
$$den_i(c) = \frac{\sum_{h \in hyper(c)} den_i(h)}{|hyper(c)|} + den(c)$$

where r is the root of WordNet hierarchy (with no hypernym), and $hyper(\cdot)$ is the set of hypernyms of a given concept. The first term is the inheritance part, normalized over all hypernyms of c in case of multiple inheritance, and the second term is the original value of density.

The resulting density values correlate significantly better with similarity. As shown in row 3 in Table 2, the correlation coefficients are about tripled on all three data sets with the new density definition den_i, and the significance of correlation is greatly improved as well (from non-correlating or

marginally correlating to strongly significantly correlating on all three data sets).

5 Using the New Definitions in Semantic Similarity Measures

In this section, we test the effectiveness of the new definitions of depth and density by using them in WordNet-based semantic similarity measures. The two similarity measures we experiment with are that of Wu and Palmer (1994) and Jiang and Conrath (1997). The first one used depth only, and the second one used both depth and density.

The task is to correlate the similarity measures with human judgment on similarity between word pairs. We use the same three data sets as in Section 3. despite the fact that *MC* is a subset of *RG* data set, we include both in order to compare with existing studies.

Correlation coefficient is calculated using *Spearman's* ρ, although results reported by some earlier studies used parametric tests such as the *Pearson Correlation Coefficient*. The reason for our choice is that the similarity scores of the word pairs in these data sets do not necessarily conform to normal distributions. Rather, we are interested in testing whether the artificial algorithms would give higher scores to pairs that are regarded closer in meaning by human judges. A non-parametric test suits better for this scenario. And this partly explains why our re-implementations of the models have lower correlation coefficients than in the original studies.

Note that there are other WordNet-based similarity measures using depth and/or density that we opt to omit for various reasons. Some of them were not designed for the particular task at hand (e.g., that of Sussna, 1993, which gives very poor correlation in this task). Others use depth of the entire WordNet hierarchy instead of individual nodes as a scaling factor (e.g., that of Leacock and Chodorow, 1998), which is unsuitable for illustrating the improvement brought about by the new depth and density definitions.

Parameterization of the weighting of depth and density is a common practice to control their individual contribution to the final similarity score (e.g., α and β in Equation (2)). Jiang and Conrath already had separate weights in their original study. In order to parameterize depth used by Wu and Palmer in their similarity measure, we also modify Equation (1) as follows:

$$sim(c_1,c_2) = \frac{2 \cdot dep^{\alpha}(c)}{len(c_1,c) + len(c_2,c) + 2 \cdot dep^{\alpha}(c)}$$

where depth is raised to the power of α to vary its contribution to the similarity score.

For a number of combinations of the weighting parameters, we report both the best performance and the averaged performance over all the parameter combinations. The latter number is meaningful in that it is a good indication of numerical stability of the parameterization. In addition, parameterization is able to generate multiple correlation coefficients, on which statistical tests can be run in order to show the significance of improvement. We use the range from 0 to 5 with step 1 for α and from 0 to 1 with step 0.1 for β.

Table 3 and 4 list the experiment results. In both models, the cumulative definition of depth dep_u consistently improve the performance of the similarity measures. In the Jiang and Conrath (1997) model, where density is applicable, the inheritance-based definition of density den_i also results in better correlation with human judgments. The optimal result is achieved when combining the new definitions of depth and density (row 4 in Table 4). For average performance, the improvement of all the new definitions over the original definitions is statistically

	Best			Average		
	MC	*RB*	*GR*	*MC*	*RB*	*GR*
dep	0.7671	0.7824	0.3773	0.7612	0.7686	0.3660
dep_u	0.7824	0.7912	0.3946	0.7798	0.7810	0.3787

Table 3: Correlation between human judgment and similarity score by Wu and Palmer (1994) using two versions of depth.

	Best			Average		
	MC	*RB*	*GR*	*MC*	*RB*	*GR*
dep, den	0.7875	0.8111	0.3720	0.7689	0.7990	0.3583
dep_u, den	0.8009	0.8181	0.3804	0.7885	0.8032	0.3669
dep, den_i	0.7882	0.8199	0.3803	0.7863	0.8102	0.3689
dep_u, den_i	0.8065	0.8202	0.3818	0.8189	0.8194	0.3715

Table 4: Correlation between human judgment and similarity score by Jiang and Conrath (1997) using different definitions of depth and density.

significant on all three data sets according to paired *t-test*.

6 Conclusions

This study explored effective uses of depth and/or density in WordNet-based similarity measures. We started by examining how well these two structural features correlate with human judgment on word pair similarities. This direct comparison showed that depth correlates with similarity only on certain value ranges, while density does not correlate with human judgment at all.

Further investigation revealed that the problem for depth lies in the simplistic representation as its ordinal integer values. The linearity in this representation fails to take into account the conflated quantity of depth in the two extreme ends of the depth spectrum. For density, a prominent issue is the information loss on specificity of WordNet concepts, which gives an inaccurate density value that is biased by the idiosyncratic constructions in densely connected regions in the hierarchy.

We then proposed new definitions of depth and density to address these issues. For depth, linearity in different value ranges is realistically reflected by projecting the depth value to its cumulative distribution function. The loss of specificity information in density, on the other hand, is corrected by allowing concepts to inherit specificity information from their parent nodes. The new definitions show significant improvement in correlation of semantic similarity given by human judges. In addition, when used in existing WordNet-based similarity measures, they consistently improve performance and numerical stability of the parameterization of the two features.

The notions of depth and density pertain to any hierarchical structure like WordNet, which suggests various extensions of this work. A natural next step of the current work is to apply the same idea to semantic taxonomies in languages other than English with available similarity judgments are also available. Extrinsic tasks using WordNet-based semantic similarity can potentially benefit from these refined notions of depth and density as well.

Acknowledgments

This study was inspired by lectures given by Professor Gerald Penn of the University of Toronto, and was financially supported by the Natural Sciences and Engineering Research Council of Canada.

References

Eneko Agirre and German Rigau. Word sense disambiguation using conceptual density. In *Proceedings of the 16th Conference on Computational Linguistics*, pages 16–22. Association for Computational Linguistics, 1996.

Martin Chodorow, Roy Byrd, and George Heidorn. Extracting semantic hierarchies from a large online dictionary. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 299–304, Chicago, Illinois, USA, 1985.

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

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, pages 406–414. ACM, 2001.

Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 305–332. 1998.

Mario Jarmasz and Stan Szpakowicz. Roget's thesaurus and semantic similarity. In *Proceedings of International Conference on Recent Advances in Natural Language Processing*, pages 212–219, Borovets, Bulgaria, 2003.

Jay Jiang and David Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *Proceedings of International Conference on Research in Computational Linguistics*, 33, 1997.

Hideki Kozima and Akira Ito. Context-sensitive measurement of word distance by adaptive scaling of a semantic space. *Recent Advances in Natural Language Processing: Selected Papers from RANLP*, 95:111–124, 1997.

Claudia Leacock and Martin Chodorow. Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 768–774, Montreal, Canada, 1998.

Goerge Miller and Walter Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet::Similarity: measuring the relatedness of concepts. In *Demonstration Papers at Human Language Technologies - North American Chapter of the Association for Computational Linguistics*, pages 38–41. Association for Computational Linguistics, 2004.

Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.

Philip Resnik. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11(11):95–130, 1999.

R. Richardson and A.F. Smeaton. Using WordNet in a knowledge-based approach to information retrieval. In *Proceedings of the BCS-IRSG Colloquium, Crewe*. Citeseer, 1995.

Herbert Rubenstein and John Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the second international conference on Information and knowledge management*, pages 67–74. ACM, 1993.

Peter Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. *Proceedings of the Twelfth European Conference on Machine Learning*, pages 491–502, 2001.

Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

Latent Vector Weighting for Word Meaning in Context

Tim Van de Cruys
RCEAL
University of Cambridge
tv234@cam.ac.uk

Thierry Poibeau
LaTTiCe, UMR8094
CNRS & ENS
thierry.poibeau@ens.fr

Anna Korhonen
Computer Laboratory & RCEAL
University of Cambridge
alk23@cam.ac.uk

Abstract

This paper presents a novel method for the computation of word meaning in context. We make use of a factorization model in which words, together with their window-based context words and their dependency relations, are linked to latent dimensions. The factorization model allows us to determine which dimensions are important for a particular context, and adapt the dependency-based feature vector of the word accordingly. The evaluation on a lexical substitution task – carried out for both English and French – indicates that our approach is able to reach better results than state-of-the-art methods in lexical substitution, while at the same time providing more accurate meaning representations.

1 Introduction

According to the distributional hypothesis of meaning (Harris, 1954), words that occur in similar contexts tend to be semantically similar. In the spirit of this by now well-known adage, numerous algorithms have sprouted up that try to capture the semantics of words by looking at their distribution in texts, and comparing those distributions in a vector space model.

Up till now, the majority of computational approaches to semantic similarity represent the meaning of a word as the aggregate of the word's contexts, and hence do not differentiate between the different senses of a word. The meaning of a word, however, is largely dependent on the particular context in which it appears. Take for example the word *work* in sentences (1) and (2).

(1) The painter's recent work is a classic example of art brut.

(2) Equal pay for equal work!

The meaning of *work* is quite different in both sentences. In sentence (1), *work* refers to the product of a creative act, viz. a painting. In sentence (2), it refers to labour carried out as a source of income. The NLP community's standard answer to the ambiguity problem has always been some flavour of word sense disambiguation (WSD), which in its standard form boils down to choosing the best-possible fit from a pre-defined sense inventory. In recent years, it has become clear that this is in fact a very hard task to solve for computers and humans alike (Ide and Wilks, 2006; Erk et al., 2009; Erk, 2010).

With these findings in mind, researchers have started looking at different methods to tackle language's ambiguity, ranging from coarser-grained sense inventories (Hovy et al., 2006) and graded sense assignment (Erk and McCarthy, 2009), over word sense induction (Schütze, 1998; Pantel and Lin, 2002; Agirre et al., 2006), to the computation of individual word meaning in context (Erk and Padó, 2008; Thater et al., 2010; Dinu and Lapata, 2010). This research inscribes itself in the same line of thought, in which the meaning disambiguation of a word is not just the assignment of a pre-defined sense; instead, the original meaning representation of a word is adapted 'on the fly', according to – and specifically tailored for – the particular context in which it appears. To be able to do so, we build a factorization model in which words, together with their window-based context words and their dependency

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1012–1022,
Edinburgh, Scotland, UK, July 27–31, 2011.

relations, are linked to latent dimensions. The factorization model allows us to determine which dimensions are important for a particular context, and adapt the dependency-based feature vector of the word accordingly. The evaluation on a lexical substitution task – carried out for both English and French – indicates that our method is able to reach better results than state-of-the-art methods in lexical substitution, while at the same time providing more accurate meaning representations.

The remainder of this paper is organized as follows. In section 2, we present some earlier work that is related to the research presented here. Section 3 describes the methodology of our method, focusing on the factorization model, and the computation of meaning in context. Section 4 presents a thorough evaluation on a lexical substitution task, both for English and French. The last section then draws conclusions, and presents a number of topics that deserve further exploration.

2 Related work

One of the best known computational models of semantic similarity is latent semantic analysis — LSA (Landauer and Dumais, 1997; Landauer et al., 1998). In LSA, a term-document matrix is created, that contains the frequency of each word in a particular document. This matrix is then decomposed into three other matrices with a mathematical factorization technique called singular value decomposition (SVD). The most important dimensions that come out of the SVD are said to represent latent semantic dimensions, according to which nouns and documents can be represented more efficiently. Our model also applies a factorization technique (albeit a different one) in order to find a reduced semantic space.

The nature of a word's context is a determining factor in the kind of the semantic similarity that is induced. A broad context window (e.g. a paragraph or document) yields broad, topical similarity, whereas a small context window yields tight, synonym-like similarity. This has lead a number of researchers (e.g. Lin (1998)) to use the dependency relations that a particular word takes part in as context features. An overview of dependency-based semantic space models is given in Padó and Lapata (2007).

A number of researchers have exploited the notion of context to differentiate between the different senses of a word in an unsupervised way (a task labeled word sense induction or WSI). Schütze (1998) proposed a context-clustering approach, in which context vectors are created for the different instances of a particular word, and those contexts are grouped into a number of clusters, representing the different senses of the word. The context vectors are represented as second-order co-occurrences (i.e. the contexts of the target word are similar if the words they in turn co-occur with are similar). Van de Cruys (2008) proposed a model for sense induction based on latent semantic dimensions. Using a factorization technique based on non-negative matrix factorization, the model induces a latent semantic space according to which both dependency features and broad contextual features are classified. Using the latent space, the model is able to discriminate between different word senses. Our approach makes use of a similar factorization model, but we extend the approach with a probabilistic framework that is able to adapt the original vector according to the context of the instance.

Recently, a number of models emerged that aim to model the individual meaning of words in context. Erk and Padó (2008, 2009) make use of selectional preferences to express the meaning of a word in context; the meaning of a word in the presence of an argument is computed by multiplying the word's vector with a vector that captures the inverse selectional preferences of the argument. Thater et al. (2009) and Thater et al. (2010) extend the approach based on selectional preferences by incorporating second-order co-occurrences in their model; their model allows first-order co-occurrences to act as a filter upon the second-order vector space, which allows for the computation of meaning in context.

Erk and Padó (2010) propose an exemplar-based approach, in which the meaning of a word in context is represented by the activated exemplars that are most similar to it. And Mitchell and Lapata (2008) propose a model for vector composition, focusing on the different functions that might be used to combine the constituent vectors. Their results indicate that a model based on pointwise multiplication achieves better results than models based on vector addition.

Finally, Dinu and Lapata (2010) propose a probabilistic framework that models the meaning of words

as a probability distribution over latent dimensions ('senses'). Contextualized meaning is then modeled as a change in the original sense distribution. The model presented in this paper bears some resemblances to their approach; however, while their approach computes the contextualized meaning directly within the latent space, our model exploits the latent space to determine the features that are important for a particular context, and adapt the original (out-of-context) dependency-based feature vector of the target word accordingly. This allows for a more precise and more distinct computation of word meaning in context. Secondly, Dinu and Lapata use window-based context features to build their latent model, while our approach combines both window-based and dependency-based features.

3 Methodology

3.1 Non-negative Matrix Factorization

Our model uses non-negative matrix factorization (Lee and Seung, 2000) in order to find latent dimensions. There are a number of reasons to prefer NMF over the better known singular value decomposition used in LSA. First of all, NMF allows us to minimize the Kullback-Leibler divergence as an objective function, whereas SVD minimizes the Euclidean distance. The Kullback-Leibler divergence is better suited for language phenomena. Minimizing the Euclidean distance requires normally distributed data, and language phenomena are typically not normally distributed. Secondly, the non-negative nature of the factorization ensures that only additive and no subtractive relations are allowed. This proves particularly useful for the extraction of semantic dimensions, so that the NMF model is able to extract much more clear-cut dimensions than an SVD model. And thirdly, the non-negative property allows the resulting model to be interpreted probabilistically, which is not straightforward with an SVD factorization.

The key idea is that a non-negative matrix $\mathbf{A}$ is factorized into two other non-negative matrices, $\mathbf{W}$ and $\mathbf{H}$

$$\mathbf{A}_{i\times j} \approx \mathbf{W}_{i\times k}\mathbf{H}_{k\times j} \tag{1}$$

where k is much smaller than i, j so that both instances and features are expressed in terms of a few components. Non-negative matrix factorization enforces the constraint that all three matrices must be non-negative, so all elements must be greater than or equal to zero.

Using the minimization of the Kullback-Leibler divergence as an objective function, we want to find the matrices $\mathbf{W}$ and $\mathbf{H}$ for which the Kullback-Leibler divergence between $\mathbf{A}$ and $\mathbf{WH}$ (the multiplication of $\mathbf{W}$ and $\mathbf{H}$) is the smallest. This factorization is carried out through the iterative application of update rules. Matrices $\mathbf{W}$ and $\mathbf{H}$ are randomly initialized, and the rules in 2 and 3 are iteratively applied – alternating between them. In each iteration, each vector is adequately normalized, so that all dimension values sum to 1.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu}\frac{\sum_i \mathbf{W}_{ia}\frac{\mathbf{A}_{i\mu}}{(\mathbf{WH})_{i\mu}}}{\sum_k \mathbf{W}_{ka}} \tag{2}$$

$$\mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia}\frac{\sum_\mu \mathbf{H}_{a\mu}\frac{\mathbf{A}_{i\mu}}{(\mathbf{WH})_{i\mu}}}{\sum_v \mathbf{H}_{av}} \tag{3}$$

3.2 Combining syntax and context words

Using an extension of non-negative matrix factorization (Van de Cruys, 2008), it is possible to jointly induce latent factors for three different modes: words, their window-based context words, and their dependency-based context features. As input to the algorithm, three matrices are constructed that capture the pairwise co-occurrence frequencies for the different modes. The first matrix contains co-occurrence frequencies of words cross-classified by dependency-based features, the second matrix contains co-occurrence frequencies of words cross-classified by words that appear in the word's context window, and the third matrix contains co-occurrence frequencies of dependency-based features cross-classified by co-occurring context words. NMF is then applied to the three matrices, and the separate factorizations are interleaved (i.e. the results of the former factorization are used to initialize the factorization of the next matrix). A graphical representation of the interleaved factorization algorithm is given in figure 1.

When the factorization is finished, the three different modes (words, window-based context words and dependency-based features) are all represented according to a limited number of latent factors.

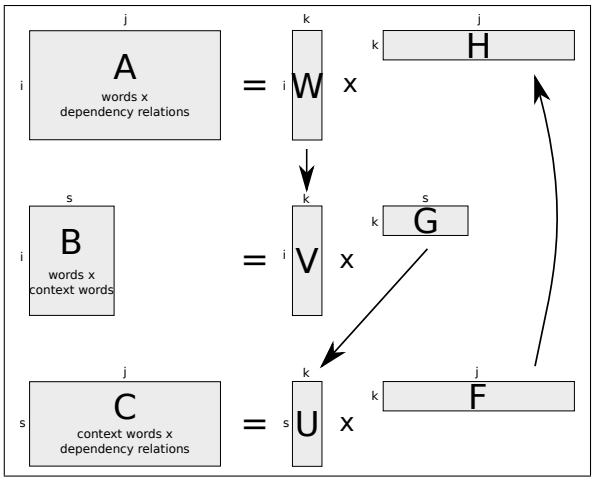


Figure 1: A graphical representation of the interleaved NMF

The factorization that comes out of the NMF model can be interpreted probabilistically (Gaussier and Goutte, 2005; Ding et al., 2008). More specifically, we can transform the factorization into a standard latent variable model of the form

$$p(w_i, d_j) = \sum_{z=1}^{K} p(z)p(w_i|z)p(d_j|z) \tag{4}$$

by introducing two $K \times K$ diagonal scaling matrices $\mathbf{X}$ and $\mathbf{Y}$, such that $\mathbf{X}_{kk} = \sum_i \mathbf{W}_{ik}$ and $\mathbf{Y}_{kk} = \sum_j \mathbf{H}_{kj}$. The factorization $\mathbf{WH}$ can then be rewritten as

$$\begin{aligned} \mathbf{WH} &= (\mathbf{WX}^{-1}\mathbf{X})(\mathbf{YY}^{-1}\mathbf{H}) \\ &= (\mathbf{WX}^{-1})(\mathbf{XY})(\mathbf{Y}^{-1}\mathbf{H}) \end{aligned} \tag{5}$$

such that $\mathbf{WX}^{-1}$ represents $p(w_i|\mathbf{z})$, $(\mathbf{Y}^{-1}\mathbf{H})^T$ represents $p(d_j|\mathbf{z})$, and $\mathbf{XY}$ represents $p(\mathbf{z})$. Using Bayes' theorem, it is now straightforward to determine $p(\mathbf{z}|w_i)$ and $p(\mathbf{z}|d_j)$.

$$p(\mathbf{z}|w_i) = \frac{p(w_i|\mathbf{z})p(\mathbf{z})}{p(w_i)} \tag{6}$$

$$p(\mathbf{z}|d_j) = \frac{p(d_j|\mathbf{z})p(\mathbf{z})}{p(d_j)} \tag{7}$$

3.3 Meaning in Context

3.3.1 Overview

Using the results of the factorization model described above, we can now adapt a word's feature vector according to the context in which it appears. Intuitively, the contextual features of the word (i.e. the window-based context words or dependency-based context features) pinpoint the important semantic dimensions of the particular instance, creating a probability distribution over latent factors. For a number of context words of a particular instance, we determine the probability distribution over latent factors given the context, $p(\mathbf{z}|C)$, as the average of the context words' probability distributions over latent factors (equation 8).

$$p(\mathbf{z}|C) = \frac{\sum_{w_i \in C} p(\mathbf{z}|w_i)}{|C|} \tag{8}$$

The probability distribution over latent factors given a number of dependency-based context features can be computed in a similar fashion, replacing w_i with d_j. Additionally, this step allows us to combine both windows-based context words and dependency-based context features in order to determine the latent probability distribution (e.g. by taking a linear combination).

The resulting probability distribution over latent factors can be interpreted as a semantic fingerprint of the passage in which the target word appears. Using this fingerprint, we can now determine a new probability distribution over dependency features given the context.

$$p(\mathbf{d}|C) = p(\mathbf{z}|C)p(\mathbf{d}|\mathbf{z}) \tag{9}$$

The last step is to weight the original probability vector of the word according to the probability vector of the dependency features given the word's context, by taking the pointwise multiplication of probability vectors $p(\mathbf{d}|w_i)$ and $p(\mathbf{d}|C)$.

$$p(\mathbf{d}|w_i, C) = p(\mathbf{d}|w_i) \cdot p(\mathbf{d}|C) \tag{10}$$

Note that this final step is a crucial one in our approach. We do not just build a model based on latent factors, but we use the latent factors to determine which of the features in the original word vector are the salient ones given a particular context. This allows us to compute an accurate adaptation of the original word vector in context.

Also note the resemblance to Mitchell and Lapata's best scoring vector composition model which, likewise, uses pointwise multiplication. However,

the model presented here has two advantages. First of all, it allows to take multiple context features into account, each of which contributes to the probability distribution over latent factors. Secondly, the target word and its features do not need to live in the same vector space (i.e. they do not need to be defined according to the same features), as the connections and the appropriate weightings are determined through the latent model.

3.3.2 Example

Let us exemplify the procedure with an example. Say we want to compute the distributionally similar words to the noun *record* in the context of example sentences (3) and (4).

(3) Jack is listening to a record.

(4) Jill updated the record.

First, we extract the context features for both instances, in this case $C_1 = \{listen^{-1}_{prep(to)}\}$ for sentence (3), and $C_2 = \{update^{-1}_{obj}\}$ for sentence (4).[1] Next, we compute $p(\mathbf{z}|C_1)$ and $p(\mathbf{z}|C_2)$ – the probability distributions over latent factors given the context – by averaging over the latent probability distributions of the individual context features.[2] Using these probability distributions over latent factors, we can now determine the probability of each dependency feature given the different contexts – $p(\mathbf{d}|C_1)$ and $p(\mathbf{d}|C_2)$.

The former step yields a general probability distribution over dependency features that tells us how likely a particular dependency feature is given the context that our target word appears in. Our last step is now to weight the original probability vector of the target word (the aggregate of dependency-based context features over all contexts of the target word) according to the new distribution given the context in which the target word appears. For the first sentence, features associated with the music sense of *record* (or more specifically, the dependency features associated with latent factors that are related to the feature $\{listen^{-1}_{prep(to)}\}$) will be emphasized, while features associated with unrelated latent factors are leveled out. For the second sentence, features that are associated with the administrative sense of *record* (dependency features associated with latent factors that are related to the feature $\{update^{-1}_{obj}\}$) are emphasized, while unrelated features are played down.

We can now return to our original matrix **A** and compute the top similar words for the two adapted vectors of *record* given the different contexts, which yields the results presented below.

1. **record**$_N$, C_1: *album*, *song*, *recording*, *track*, *cd*

2. **record**$_N$, C_2: *file*, *datum*, *document*, *database*, *list*

4 Evaluation

In this section, we present a thorough evaluation of the method described above, and compare it with related methods for meaning computation in context. In order to test the applicability of the method to multiple languages, we present evaluation results for both English and French.

4.1 Datasets

For English, we make use of the SEMEVAL 2007 English Lexical Substitution task (McCarthy and Navigli, 2007; McCarthy and Navigli, 2009). The task's goal is to find suitable substitutes for a target word in a particular context. The complete data set contains 200 target words (about 50 for each part of speech, viz. nouns, verbs, adjectives, and adverbs). Each target word occurs in 10 different sentences, which yields a total of 2000 sentences. Five annotators provided suitable substitutes for each target word in the different contexts.

For French, we developed a small-scale lexical substitution task ourselves, closely following the guidelines of the original English task. We manually selected 10 ambiguous French nouns, and for each noun we selected 10 different sentences from the FRWAC corpus (Baroni et al., 2009). Four different native French speakers were then asked to provide suitable substitutes for the nouns in context.[3]

[1] In this example we use dependency features, but the computations are similar for window-based context words.

[2] In this case, the sets of context features contain only one item, so the average probability distribution of the sets is just the latent probability distribution of their respective item.

[3] The task is provided as supplementary material to this paper; it is also available from the first author's website.

4.2 Implementational details

The model for English has been trained on part of the UKWaC corpus (Baroni et al., 2009), covering about 500M words. The corpus has been part of speech tagged and lemmatized with Stanford Part-Of-Speech Tagger (Toutanova and Manning, 2000; Toutanova et al., 2003), and parsed with MaltParser (Nivre et al., 2006) trained on sections 2-21 of the Wall Street Journal section of the Penn Treebank extended with about 4000 questions from the QuestionBank[4], so that dependency triples could be extracted. The sentences of the English lexical substitution task have been tagged, lemmatized and parsed in the same way. The model for French has been trained on the French version of Wikipedia ($\pm$ 100M words), parsed with the FRMG parser (Villemonte de La Clergerie, 2010) for French.

For English, we built different models for each part of speech (nouns, verbs, adjectives and adverbs), which yields four models in total. For each model, the matrices needed for our interleaved NMF factorization are extracted from the corpus. The noun model, for example, was built using 5K nouns, 80K dependency relations, and 2K context words[5] (excluding stop words) with highest frequency in the training set, which yields matrices of 5K nouns $\times$ 80K dependency relations, 5K nouns $\times$ 2K context words, and 80K dependency relations $\times$ 2K context words. The models for the three other parts of speech were constructed in a similar vein. For French, we only constructed a model for nouns, as our lexical substitution task for French is limited to this part of speech.

The interleaved NMF model was carried out using $K = 600$ (the number of factorized dimensions in the model), and applying 100 iterations.[6] The interleaved NMF algorithm was implemented in Matlab; the preprocessing scripts and scripts for vector computation in context were written in Python. Cosine was used as a similarity measure.

[4] http://maltparser.org/mco/english_parser/engmalt.html

[5] We used a fairly large, paragraph-like window of four sentences.

[6] We experimented with different values (in the range 300–1500) for K, but the models did not seem to improve much beyond $K = 600$; hence, we stuck with 600 factors, due to speed and memory advantages of a lower number of factors.

4.3 Measures

Up till now, most researchers have interpreted the lexical substitution task as a ranking problem, in which the possible substitutes are given beforehand and the goal is to rank the substitutes according to their suitability in a particular context, so that sound substitutes are given a higher rank than their non-suitable counterparts. This means that all possible substitutes for a given target word (extracted from the gold standard) are lumped together, and the system then has to produce a ranking for the complete set of substitutes.

We also adopt this approach in our evaluation framework, but we complement it with the original evaluation measures of the lexical substitution task, in which the system is not given a list of possible substitutes beforehand, but has to come up with the suitable candidates itself. This is a much harder task, but we believe that such an approach is more compelling in assessing the system's ability to induce a proper meaning representation for word usage in context. We coin the former approach *paraphrase ranking*, and the latter one *paraphrase induction*. In the next paragraphs, we will describe the actual evaluation measures that have been used for both approaches.

Paraphrase ranking Following Dinu and Lapata (2010), we compare the ranking produced by our model with the gold standard ranking using Kendall's τ_b (which is adjusted for ties). For reasons of comparison, we also compute general average precision (GAP, Kishida (2005)), which was used by Erk and Padó (2010) and Thater et al. (2010) to evaluate their rankings. Differences between models are tested for significance using stratified shuffling (Yeh, 2000), using a standard number of 10000 iterations.

We compare the results for paraphrase ranking to two different baselines. The first baseline is a random one, in which the gold standard is compared to an arbitrary ranking. The second baseline is a dependency-based vector space model that does not take the context of the particular instance into account (and thus returns the same ranking for each instance of the target word). This is a fairly competitive baseline, as noted by other researchers (Erk and Padó, 2008; Thater et al., 2009; Dinu and Lapata, 2010).

Paraphrase induction To evaluate the system's ability to come up with suitable substitutes from scratch, we use the measures designed to evaluate systems that took part in the original English lexical substitution task (McCarthy and Navigli, 2007). Two different measures were used, which were coined *best* and *out-of-ten (oot)*. The strict *best* measure allows the system to give as many candidate substitutes as it considers appropriate, but the credit for each correct substitute is divided by the total number of guesses. Recall is then calculated as the average annotator response frequency of substitutes found by the system over all items T.

$$R_{best} = \frac{\sum_{s \in M \cap G} f(s)}{|M| \cdot \sum_{s \in G} f(s)} \quad (11)$$

where M is the system's candidate list[7], G is the gold-standard data, and $f(s)$ is the annotator response frequency of the candidate.

The out-of-ten measure is more liberal; it allows the system to give up to ten substitutes, and the credit for each correct substitute is not divided by the total number of guesses. The more liberal measure was introduced to account for the fact that the lexical substitution task's gold standard is susceptible to a considerate amount of variation, and there is only a limited number of annotators.

$$P_{10} = \frac{\sum_{s \in M \cap G} f(s)}{\sum_{s \in G} f(s)} \quad (12)$$

where M is the system's list of 10 candidates, and G and $f(s)$ are the same as above. Because we only use the best guess with R_{best}, the two measures are exactly the same except for the number of candidates M.

4.4 Results

4.4.1 English

Table 1 presents the paraphrase ranking results of our approach, comparing them to the two baselines and to a number of previous approaches to meaning computation in context.

The first two models represent our baselines. The first baseline is the random baseline, where the candidate substitutes are ranked randomly (τ_b close to zero indicates that there is no correlation). The second baseline is a standard dependency-based vector space model, which yields the same ranking for all instances of a target word. Note that the second baseline is a rather competitive one.

model	τ_b	GAP
random	-0.61	29.98
vector$_{dep}$	16.57	45.08
EP09	–	32.2 ▼
EP10	–	39.9 ▼
TFP	–	45.94▼
DL	16.56	41.68
NMF$_{context}$	20.64**	47.60**
NMF$_{dep}$	22.49**	48.97**
NMF$_{c+d}$	**22.59****	**49.02****

Table 1: Kendall's τ_b and GAP paraphrase ranking scores for the English lexical substitution task. Scores marked with '▼' are copied from the authors' respective papers. Scores marked with '**' are statistically significant with $p < 0.01$ compared to the second baseline.

The next four models represent previous approaches to meaning computation in context. EP09 is Erk and Pado's (2009) selectional preference approach; EP10 is Erk and Pado's (2010) exemplar-based approach; TFP stands for Thater et al.'s (2010) approach; and DL is Dinu and Lapata's (2010) latent modeling approach. The results are reproduced from their respective papers, except for Dinu and Lapata's approach, which we reimplemented ourselves.[8] Note that the reproduced results (EP09, EP10 and TFP) are not entirely comparable, because the authors only use a subset of the lexical substitution task.

The last three models are instantiations of our approach: NMF$_{context}$ is a model that uses window-based context features, NMF$_{dep}$ is a model that uses dependency-based context features, and NMF$_{c+d}$ is a model that uses a linear combination of window-based and dependency-based context features, giving equal weight to both.

The three instantiations of our approach reach better results than all previous approaches. Moreover, our approach is the only one able to significantly

[7] In our evaluations, we calculate *best* using the system's best guess only, so the candidate list contains only one item.

[8] The original paper reports a slightly lower τ_b of 16.01 for their best scoring model.

beat our second (competitive) baseline of a standard dependency-based vector model. Comparing our three instantiations, the model that combines window-based context and dependency-based context scores best, closely followed by the dependency-based model. The model that only uses window-based context gets the lowest score of the three, but is still fairly competitive compared to the previous approaches. The differences between the models are statistically significant ($p < 0.01$), except for the difference between NMF$_{dep}$ and NMF$_{c+d}$.

model	n	v	a	r
vector$_{dep}$	15.85	11.68	16.71	25.29
NMF$_{context}$	20.58	16.24	21.00	27.22
NMF$_{dep}$	21.96	17.33	**24.57**	28.16
NMF$_{c+d}$	**22.68**	**17.47**	23.84	**28.66**

Table 2: Kendall's τ_b paraphrase ranking scores for the English lexical substitution task across different parts of speech

Table 2 shows the performance of the three model instantiations on paraphrase ranking across different parts of speech. The results largely confirm tendencies reported by other researchers (cfr. Dinu and Lapata (2010)), viz. that verbs are the most difficult, followed by nouns and adjectives. These parts of speech also benefit the most from the use of a contextualized model. Adverbs are easier, but there is less to be gained from using contextualized models.

model	R_{best}	P_{10}
vector$_{dep}$	8.78	30.21
DL	1.06	7.59
KU	20.65	46.15
IRST2	20.33	68.90
NMF$_{context}$	8.81	**30.49**
NMF$_{dep}$	7.73	26.92
NMF$_{c+d}$	**8.96**	29.26

Table 3: R_{best} and P_{10} paraphrase induction scores for the English lexical substitution task

Table 3 shows the performance of the different models on the paraphrase induction task. Note once again that our baseline vector$_{dep}$ – a simple dependency-based vector space model – is a highly competitive one. NMF$_{context}$ and NMF$_{c+d}$ are able to reach marginally better results, but the differences are not statistically significant. However, all of our models are able to reach much better results than Dinu and Lapata's approach. The results indicate that our approach, after vector adaptation in context, is still able to provide accurate similarity calculations across the complete word space. While other algorithms are able to rank candidate substitutes at the expense of accurate similarity calculations, our approach is able to do both. This is one of the important advantages of our approach.

For reasons of comparison, we also included the scores of the best performing models that participated in the SEMEVAL 2007 lexical substitution task (KU (Yuret, 2007) and IRST2 (Giuliano et al., 2007), which got the best scores for R_{best} and P_{10}, respectively). These models reach better scores compared to our models. Note, however, that all participants of the SEMEVAL 2007 lexical substitution task relied on a predefined sense inventory (i.e. WordNet, or a machine readable thesaurus). Our system, on the other hand, induces paraphrases in a fully unsupervised way. To our knowledge, this is the first time a fully unsupervised system is tested on the paraphrase induction task.

model	n	v	a	r
vector$_{dep}$	31.66	23.53	29.91	38.43
NMF$_{context}$	**33.73****	25.21*	28.58	36.45
NMF$_{dep}$	31.40	25.97**	20.56	31.48
NMF$_{c+d}$	33.37*	**25.99****	24.20	35.81

Table 4: P_{10} paraphrase induction scores for the English lexical substitution task across different parts of speech. Scores marked with '**' and '*' are statistically significant with respectively $p < 0.01$ and $p < 0.05$ compared to the baseline.

Table 4 presents the results for paraphrase induction (*oot*) across the different parts of speech. The results indicate that paraphrase induction works best for nouns and verbs, with statistically significant improvements over the baseline. The differences among the models themselves are not significant. Adjectives and adverbs yield lower scores, indicating that their

contextualization yields less precise vectors for meaning computation. Note, however, that the NMF$_{context}$ model is still quite apt for meaning computation, yielding results that are only slightly lower than the dependency-based vector space model.

4.4.2 French

This section presents the results on the French lexical substitution task. Table 5 presents the results for paraphrase ranking, while table 6 shows the models' performance on the paraphrase induction task.

model	Kendall's τ_b	GAP
vector$_{dep}$	7.79	36.46
DL	17.99	41.73
NMF$_{context}$	**18.63**	**44.96**
NMF$_{dep}$	17.15	44.66
NMF$_{c+d}$	18.40	43.14

Table 5: Kendall's τ_b and GAP paraphrase ranking scores for the French lexical substitution task

The results for paraphrase ranking in French (table 5) show similar tendencies as the results for English: all of our models are able to improve significantly over the dependency-based vector space baseline. Note, however, thar our models generally score a bit lower compared to the English results. This drop in performance is not present for Dinu and Lapata's model. The difference might be due to the difference in corpora size: for the method to operate at full power, we need to make a good estimate of the co-occurrences of three modes (words, window-based context words and dependency-based features), and thus our methods requires a significant amount of data. Nevertheless, our approach still yields the best results, with NMF$_{context}$ as the best scoring model.

Finally, the results for paraphrase induction in French (table 6) interestingly show a significant and large improvement over the baseline. The improvements indicate once again that the models are able to carry out precise similarity computations over the whole word space, while at the same time providing an adequately adapted contextualized meaning vector. Dinu and Lapata's model, which performs similarity calculations in the latent space, is not able to provide accurate word vectors, and thus perform worse at the paraphrase induction task.

model	R_{best}	P_{10}
vector$_{dep}$	6.38	24.43
DL	0.50	5.34
NMF$_{context}$	**10.71**	31.42
NMF$_{dep}$	9.65	28.52
NMF$_{c+d}$	10.64	**35.32**

Table 6: R_{best} and P_{10} paraphrase induction scores for the French lexical substitution task

5 Conclusion

In this paper, we presented a novel method for the modeling of word meaning in context. We make use of a factorization model based on non-negative matrix factorization, in which words, together with their window-based context words and their dependency relations, are linked to latent dimensions. The factorization model allows us to determine which particular dimensions are important for a target word in a particular context. A key feature of the algorithm is that we adapt the original dependency-based feature vector of the target word through the latent semantic space. By doing so, our model is able to make accurate similarity calculations for word meaning in context across the whole word space. Our evaluation shows that the approach presented here is able to improve upon the state-of-the art performance on paraphrase ranking. Moreover, our approach scores well for both paraphrase ranking and paraphrase induction, whereas previous approaches only seem capable of improving performance on the former task at the expense of the latter.

During our research, a number of topics surfaced that we consider worth exploring in the future. First of all, we would like to further investigate the optimal configuration for combining window-based and dependency-based contexts. At the moment, the performance of the combined model does not yield a uniform picture. The results might improve further if window-based context and dependency-based context are combined in an optimal way. Secondly, we would like to subject our approach to further evaluation, in particular on a number of different evaluation tasks, such as semantic compositionality. And thirdly, we would like to transfer the general idea of the approach presented in this paper to a tensor-

based framework (which is able to capture the multi-way co-occurrences of words, together with their window-based and dependency-based context features, in a natural way) and investigate whether such a framework proves beneficial for the modeling of word meaning in context.

Acknowledgements

The work reported in this paper was funded by the Isaac Newton Trust (Cambridge, UK), the EU FP7 project 'PANACEA', the EPSRC grant EP/G051070/1 and the Royal Society (UK).

References

Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006. Two graph-based algorithms for state-of-the-art wsd. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP) Conference*, pages 585–593, Sydney, Australia.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Chris Ding, Tao Li, and Wei Peng. 2008. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927.

Georgiana Dinu and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, Cambridge, MA, October.

Katrin Erk and Diana McCarthy. 2009. Graded word sense assignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 440–449, Suntec, Singapore.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906, Waikiki, Hawaii, USA.

Katrin Erk and Sebastian Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 57–65, Athens, Greece.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 92–97, Uppsala, Sweden.

Katrin Erk, Diana McCarthy, and Nicholas Gaylord. 2009. Investigations on word senses and word usages. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 10–18.

Katrin Erk. 2010. What is word meaning, really? (and how can distributional models help us describe it?). In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 17–26.

Eric Gaussier and Cyril Goutte. 2005. Relation between PLSA and NMF and implications. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602, Salvador, Brazil.

Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava. 2007. Fbk-irst: Lexical substitution task exploiting domain and syntagmatic coherence. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, pages 145–148.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 57–60, New York, New York, USA.

Nancy Ide and Yorick Wilks. 2006. Making Sense About Sense. In *Word Sense Disambiguation: Algorithms And Applications*, chapter 3. Springer, Dordrecht.

Kazuaki Kishida. 2005. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. Technical report, National Institute of Informatics.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychology Review*, 104:211–240.

Thomas Landauer, Peter Foltz, and Darrell Laham. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:295–284.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL98), Volume 2*, pages 768–774, Montreal, Quebec, Canada.

Diana McCarthy and Roberto Navigli. 2007. SemEval-2007 task 10: English lexical substitution task. In

Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 48–53.

Diana McCarthy and Roberto Navigli. 2009. The English lexical substitution task. *Language resources and evaluation*, 43(2):139–159.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *proceedings of ACL-08: HLT*, pages 236–244.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619, Edmonton, Alberta, Canada.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal. 2009. Ranking paraphrases in context. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 44–47, Suntec, Singapore.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 948–957, Uppsala, Sweden.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 63–70.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Tim Van de Cruys. 2008. Using three way data for word sense discrimination. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 929–936, Manchester.

Eric Villemonte de La Clergerie. 2010. Building factorized TAGs with meta-grammars. In *Proceedings of the 10th International Conference on Tree Adjoining Grammars and Related Formalisms (TAG+10)*, pages 111–118, New Haven, Connecticut, USA.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics*, pages 947–953, Saarbrücken, Germany.

Deniz Yuret. 2007. Ku: Word sense disambiguation by substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, pages 207–213.

Hierarchical Verb Clustering Using Graph Factorization

Lin Sun and Anna Korhonen
University of Cambridge, Computer Laboratory
15 JJ Thomson Avenue, Cambridge CB3 0GD, UK
ls418,alk23@cl.cam.ac.uk

Abstract

Most previous research on verb clustering has focussed on acquiring flat classifications from corpus data, although many manually built classifications are taxonomic in nature. Also Natural Language Processing (NLP) applications benefit from taxonomic classifications because they vary in terms of the granularity they require from a classification. We introduce a new clustering method called Hierarchical Graph Factorization Clustering (HGFC) and extend it so that it is optimal for the task. Our results show that HGFC outperforms the frequently used agglomerative clustering on a hierarchical test set extracted from VerbNet, and that it yields state-of-the-art performance also on a flat test set. We demonstrate how the method can be used to acquire novel classifications as well as to extend existing ones on the basis of some prior knowledge about the classification.

1 Introduction

A variety of verb classifications have been built to support NLP tasks. These include syntactic and semantic classifications, as well as ones which integrate aspects of both (Grishman et al., 1994; Miller, 1995; Baker et al., 1998; Palmer et al., 2005; Kipper, 2005; Hovy et al., 2006). Classifications which integrate a wide range of linguistic properties can be particularly useful for tasks suffering from data sparseness. One such classification is the taxonomy of English verbs proposed by Levin (1993) which is based on shared (morpho-)syntactic and semantic properties of verbs. Levin's taxonomy or its extended version in VerbNet (Kipper, 2005) has proved helpful for various NLP application tasks, including e.g. parsing, word sense disambiguation, semantic role labeling, information extraction, question-answering, and machine translation (Swier and Stevenson, 2004; Dang, 2004; Shi and Mihalcea, 2005; Zapirain et al., 2008).

Because verbs change their meaning and behaviour across domains, it is important to be able to tune existing classifications as well to build novel ones in a cost-effective manner, when required. In recent years, a variety of approaches have been proposed for automatic induction of Levin style classes from corpus data which could be used for this purpose (Schulte im Walde, 2006; Joanis et al., 2008; Sun et al., 2008; Li and Brew, 2008; Korhonen et al., 2008; Ó Séaghdha and Copestake, 2008; Vlachos et al., 2009). The best of such approaches have yielded promising results. However, they have mostly focussed on acquiring and evaluating flat classifications. Levin's classification is not flat, but taxonomic in nature, which is practical for NLP purposes since applications differ in terms of the granularity they require from a classification.

In this paper, we experiment with hierarchical Levin-style clustering. We adopt as our baseline method a well-known hierarchical method – agglomerative clustering (AGG) – which has been previously used to acquire flat Levin-style classifications (Stevenson and Joanis, 2003) as well as hierarchical verb classifications not based on Levin (Ferrer, 2004; Schulte im Walde, 2008). The method has also been popular in the related task of noun clus-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1023–1033,
Edinburgh, Scotland, UK, July 27–31, 2011.

tering (Ushioda, 1996; Matsuo et al., 2006; Bassiou and Kotropoulos, 2011).

We introduce then a new method called Hierarchical Graph Factorization Clustering (HGFC) (Yu et al., 2006). This graph-based, probabilistic clustering algorithm has some clear advantages over AGG (e.g. it delays the decision on a verb's cluster membership at any level until a full graph is available, minimising the problem of error propagation) and it has been shown to perform better than several other hierarchical clustering methods in recent comparisons (Yu et al., 2006). The method has been applied to the identification of social network communities (Lin et al., 2008), but has not been used (to the best of our knowledge) in NLP before.

We modify HGFC with a new tree extraction algorithm which ensures a more consistent result, and we propose two novel extensions to it. The first is a method for automatically determining the tree structure (i.e. number of clusters to be produced for each level of the hierarchy). This avoids the need to predetermine the number of clusters manually. The second is addition of soft constraints to guide the clustering performance (Vlachos et al., 2009). This is useful for situations where a partial (e.g. a flat) verb classification is available and the goal is to extend it.

Adopting a set of lexical and syntactic features which have performed well in previous works, we compare the performance of the two methods on test sets extracted from Levin and VerbNet. When evaluated on a flat clustering task, HGFC outperforms AGG and performs very similarly with the best flat clustering method reported on the same test set (Sun and Korhonen, 2009). When evaluated on a hierarchical task, HGFC performs considerably better than AGG at all levels of gold standard classification. The constrained version of HGFC performs the best, as expected, demonstrating the usefulness of soft constraints for extending partial classifications.

Our qualitative analysis shows that HGFC is capable of detecting novel information not included in our gold standards. The unconstrained version can be used to acquire novel classifications from scratch while the constrained version can be used to extend existing ones with additional class members, classes and levels of hierarchy.

2 Target classification and test sets

The taxonomy of Levin (1993) groups English verbs (e.g. *break, fracture, rip*) into classes (e.g. 45.1 *Break* verbs) on the basis of their shared meaning components and (morpho-)syntactic behaviour, defined in terms of diathesis alternations (e.g. the causative/inchoative alternation, where an NP frame alternates with an intransitive frame: *Tony broke the window* $\leftrightarrow$ *The window broke*). It classifies over 3000 verbs in 57 top level classes, some of which divide further into subclasses. The extended version of the taxonomy in VerbNet (Kipper, 2005) classifies 5757 verbs. Its 5 level taxonomy includes 101 top level and 369 subclasses. We used three gold standards (and corresponding test sets) extracted from these resources in our experiments:

T1: The first gold standard is a flat gold standard which includes 13 classes appearing in Levin's original taxonomy (Stevenson and Joanis, 2003). We included this small gold standard in our experiments so that we could compare the flat version of our method against previously published methods. Following Stevenson and Joanis (2003), we selected 20 verbs from each class which occur at least 100 times in our corpus. This gave us 260 verbs in total.

T2: The second gold standard is a large, hierarchical gold standard which we extracted from VerbNet as follows: 1) We removed all the verbs that have less than 1000 occurrences in our corpus. 2) In order to minimise the problem of polysemy, we assigned each verb to the class which, according to VerbNet, corresponds to its predominant sense in WordNet (Miller, 1995). 3) In order to minimise the sparse data problem with very fine-grained classes, we converted the resulting classification into a 3-level representation so that the classes at the 4th and 5th level were combined. For example, the sub-classes of *Declare* verbs (numbered as 29.4.1.1.$\{$1,2,3$\}$) were combined into 29.4.1. 4) The classes that have fewer than 5 members were discarded. The total number of verb senses in the resulting gold standard is 1750, which is 33.2% of the verbs in VerbNet. T2 has 51 top level, 117 second level, and 133 third level classes.

T3: The third gold standard is a subset of T2 where singular classes (top level classes which do not divide into subclasses) are removed. This gold

standard was constructed to enable proper evaluation of the constrained version of HGFC (introduced in the following section) where we want to compare the impact of constraints across several levels of classification. T3 provides classification of 357 verbs into 11 top level, 14 second level, and 32 third level classes.

For each verb appearing in T1-T3, we extracted all the occurrences (up to 10,000) from the British National Corpus (Leech, 1992) and North American News Text Corpus (Graff, 1995).

3 Method

3.1 Features and feature extraction

Previous works on Levin style verb classification have investigated optimal features for this task (Stevenson and Joanis, 2003; Li and Brew, 2008; Sun and Korhonen, 2009)). We adopt for our experiments a set of features which have performed well in recent verb clustering works:

A: Subcategorization frames (SCFs) and their relative frequencies with individual verbs.

B: A with SCFs parameterized for prepositions.

C: B with SCFs parameterized for subjects appearing in grammatical relations associated with the verb in parsed data.

D: B with SCFs parameterized for objects appearing in grammatical relations associated with the verb in parsed data.

These features are purely syntactic. Although semantic features – verb selectional preferences – proved the best (when used in combination with syntactic features) in the recent work of Sun and Korhonen (2009), we left such features for future work because we noticed that different levels of classification are likely to require semantic features at different granularities.

We extracted the syntactic features using the system of Preiss et al. (2007). The system tags, lemmatizes and parses corpus data using the RASP (Robust Accurate Statistical Parsing toolkit (Briscoe et al., 2006)), and on the basis of the resulting grammatical relations, assigns each occurrence of a verb as a member of one of the 168 verbal SCFs. We parameterized the SCFs as described above using the information provided by the system.

3.2 Clustering

We introduce the agglomerative clustering (AGG) and Hierarchical Graph Factorization Clustering (HGFC) methods in the following two subsections, respectively. The subsequent two subsections present our extensions to HGFC: (i) automatically determining the cluster structure and (ii) adding soft constraints to guide clustering performance.

3.2.1 Agglomerative clustering

AGG is a method which treats each verb as a singleton cluster and then successively merges two closest clusters until all the clusters have been merged into one. We used the SciPy's implementation (Oliphant, 2007) of the algorithm. The cluster distance is measured using linkage criteria. We experimented with four commonly used linkage criteria: Single, Average, Complete and Ward's (Ward Jr., 1963). Ward's criterion performed the best and was used in all the experiments in this paper. It measures the increase in variance after two clusters are merged. The output of AGG tends to have excessive number of levels. Cut-based methods (Wu and Leahy, 1993; Shi and Malik, 2000) are frequently applied to extract a simplified view. We followed previous verb clustering works and cut the AGG hierarchy manually.

AGG suffers from two problems. The first is error propagation. When a verb is misclassified at a lower level, the error propagates to all the upper levels. The second is local pairwise merging, i.e. the fact that only two clusters can be combined at any level. For example, in order to group clusters representing Levin classes 9.1, 9.2 and 9.3 into a single cluster representing class 9, the method has to produce intermediate clusters, e.g. 9.{1,2} and 9.3. Such clusters do not always have a semantic interpretation. Although they can be removed using a cut-based method, this requires a pre-defined cut-off value which is difficult to set (Stevenson and Joanis, 2003). In addition, a significant amount of information is lost in pair-wise clustering. In the above example, only the clusters 9.{1,2} and 9.3 are considered, while alternative clusters 9.{1,3} and 9.2 are ignored. Ideally, information about all the possible intermediate clusters should be aggregated, but this is intractable in practice.

3.2.2 Hierarchical Graph Factorization Clustering

Our new method HGFC derives a probabilistic bipartite graph from the similarity matrix (Yu et al., 2006). The local and global clustering structures are learned via the random walk properties of the graph.

The method does not suffer from the above problems with AGG. Firstly, there is no error propagation because the decision on a verb's membership at any level is delayed until the full bipartite graph is available and until a tree structure can be extracted from it by aggregating probabilistic information from all the levels. Secondly, the bipartite graph enables the construction of a hierarchical structure without any intermediate classes. For example, we can group classes 9.{1,2,3} directly into class 9.

We use HGFC with the distributional similarity measure Jensen-Shannon Divergence ($d_{js}(v, v')$). Given a set of verbs, $V = \{v_n\}_{n=1}^{N}$, we compute a similarity matrix W where $W_{ij} = \exp(-d_{js}(v_1, v_2))$. W can be encoded by a undirected graph G (Figure 1(a)), where the verbs are mapped to vertices and the W_{ij} is the edge weight between vertices i and j.

The graph G and the cluster structure can be represented by a bipartite graph $K(V, U)$. V are the vertices on G. $U = \{u_p\}_{p=1}^{m}$ represent the hidden m clusters. For example, looking at Figure 1(b), V on G can be grouped into three clusters u_1, u_2 and u_3. The matrix B denotes the $n \times m$ adjacency matrix, with b_{ip} being the connection weight between the vertex v_i and the cluster u_p. Thus, B represents the connections between clusters at an upper and lower level of clustering. A flat clustering algorithm can be induced by computing B.

The bipartite graph K also induces a similarity (W') between v_i and v_j: $w'_{ij} = \sum_{p=1}^{m} \frac{b_{ip} b_{jp}}{\lambda_p} = (B\Lambda^{-1}B^T)_{ij}$ where $\Lambda = \text{diag}(\lambda_1, ..., \lambda_m)$. Therefore, B can be found by approximating the similarity matrix W of G using W' derived from K. Given a distance function ζ between two similarity matrices, B approximates W by minimizing the cost function $\zeta(W, B\Lambda^{-1}B^T)$. The coupling between B and Λ is removed by setting $H = B\Lambda^{-1}$:

$$\min_{H,\Lambda} \zeta(W, H\Lambda H^T), s.t. \sum_{i=1}^{n} h_{ip} = 1 \quad (1)$$

We use the divergence distance: $\zeta(X, Y) = \sum_{ij}(x_{ij} \log \frac{x_{ij}}{y_{ij}} - x_{ij} + y_{ij})$. Yu et al. (2006) showed that this cost function is non-increasing under the update rule:

$$\tilde{h}_{ip} \propto h_{ip} \sum_j \frac{w_{ij}}{(H\Lambda H^T)_{ij}} \lambda_p h_{jp} \text{ s.t.} \sum_i \tilde{h}_{ip} = 1 \quad (2)$$

$$\tilde{\lambda}_p \propto \lambda_p \sum_j \frac{w_{ij}}{(H\Lambda H^T)_{ij}} h_{ip} h_{jp} \text{ s.t.} \sum_p \tilde{\lambda}_p = \sum_{ij} w_{ij} \quad (3)$$

w_{ij} can be interpreted as the probability of the direct transition between v_i and v_j: $w_{ij} = p(v_i, v_j)$, when $\sum_{ij} w_{ij} = 1$. b_{ip} can be interpreted as:

$$\begin{aligned} p(u_p, u_q) &= p(u_p)p(u_p|u_q) = \sum_{i=1}^{n} \frac{b_{ip} b_{iq}}{d_i} \\ &= (B^T D^{-1} B)_{pq} \quad (4) \\ D &= \text{diag}(d_1, ..., d_n) \text{ where } d_i = \sum_{p=0}^{m} b_{ip} \end{aligned}$$

$p(u_p, u_q)$ is the similarity between the clusters. It takes into account of a weighted average of contributions from all the data. This is different from the linkage method where only the data from two clusters are considered.

Given the cluster similarity $p(u_p, u_q)$, we can construct a new graph G_1 (Figure 1(d)) with the clusters U as vertices. The cluster algorithm can be applied again (Figure 1(e)). This process can go on iteratively, leading to a hierarchical graph.

Algorithm 1 HGFC algorithm (Yu et al., 2006)

Require: N verbs V, number of clusters m_l for L levels
  Compute the similarity matrix W_0 from V
  Build the graph G_0 from W_0 , and $m_0 \leftarrow n$
  for $l = 1, 2$ to L **do**
    Factorize G_{l-1} to obtain bipartite graph K_l with the adjacency matrix B_l (eq. 1, 2 and 3)
    Build a graph G_l with similarity matrix $W_l = B_l^T D_l^{-1} B_l$ according to equation 4
  end for
  return $B_L, B_{L-1}...B_1$

Additional steps need to be performed in order to extract a tree from the hierarchical graph. Yu et al. (2006) performs the extraction via a propagation of probabilities from the bottom level clusters. For a verb v_i, the probability of assigning it to cluster $v_p^{(l)}$ at level l is given by:

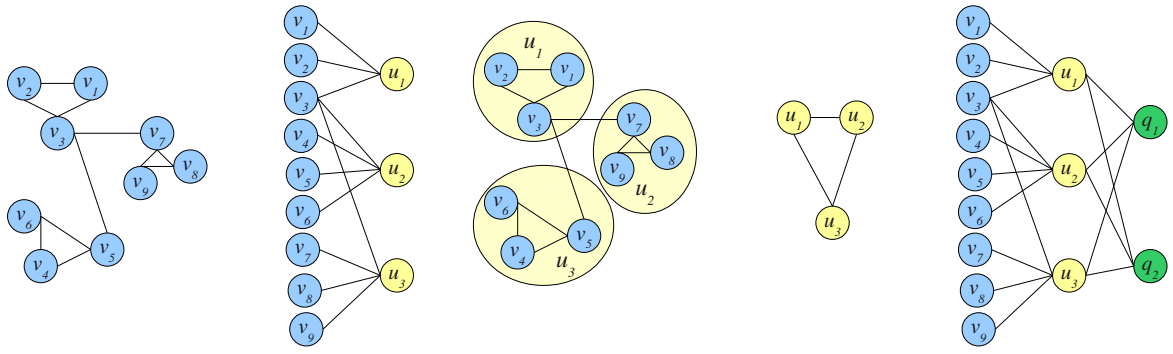


Figure 1: (a) An undirected graph G representing the similarity matrix; (b) The bipartite graph showing three clusters on G; (c) The induced clusters U; (d) The new graph G_1 over clusters U; (e) The new bipartite graph over G_1

$$p(v_p^{(l)}|v_i) = \sum_{V_{l-1}} ... \sum_{V_1} p(v_p^{(l)}|v^{(l-1)})...p(v^{(1)}|v_i)$$
$$= (D_1^{(-1)} B_1 D_2^{-1} B_2 D_3^{-1} B_3 ... D_l^{-1} B_l)_{ip} \quad (5)$$

This method might not extract a consistent tree structure, because the cluster membership at the lower level does not constrain the upper level membership. This prevented us from extracting a Levin style hierarchical classification in our initial experiments. For example, where two verbs were grouped together at a lower level, they could belong to separate clusters at an upper level. We therefore propose a new tree extraction algorithm (Algorithm 2).

The new algorithm starts from the top level bipartite graph, and generates consistent labels for each level by taking into account of the tree constraints set at upper levels.

Algorithm 2 Tree extraction algorithm for HGFC

Require: Given N, (B^l, m_l) on each level for L levels
On the top level L, collect the labels T^L (eq. 5)
Define C to be a $(m_{L-1} \times m_L)$ zero matrix, $C_{ij} \leftarrow 1$, where $i, j = \arg\max_{i,j}\{B_{ij}^L\}$
for $l = L-1$ to 1 **do**
  for $i = 1$ to N **do**
    Compute $p(v_p^l|v_i)$ for each cluster p (eq. 5)
    $t_i^l = \text{argmax}_p\{p(v_p^l|v_i)|p = 1...m_l, C_{pt_i^{l+1}} \neq 0\}$
  end for
  Redefine C to be a $(m_{l-1} \times m_l)$ zero matrix, $C_{ij} \leftarrow 1$, where $i, j = \arg\max_{i,j}\{B_{ij}^l\}$
end for
return Tree consistent labels $T^L, T^{L-1}...T^1$

3.2.3 Automatically determining the number of clusters for HGFC

HGFC needs the number of levels and clusters at each level as input. However, this information is not always available (e.g. when the goal is to actually learn this information automatically). We therefore propose a method for inferring the cluster structure from data. As shown in figure 1, a similarity matrix W models one-hop transitions that follow the links from vertices to neighbors. A walker can also go to other vertices via multi-hop transitions. According to the chain rule of the Markov process, the multi-hop transitions indicate a decaying similarity function on the graph (Yu et al., 2006). After t transitions, the similarity matrix (W_t) becomes:

$$W_t = W_{t-1} D_0^{-1} W_0$$

Yu et al. (2006) proved the correspondence between the HGFC levels (l) and the random walk time: $t = 2^{l-1}$. So the vertices at level l induce a similarity matrix of verbs after t-hop transitions. The decaying similarity function captures the different scales of clustering structure in the data (Azran and Ghahramani, 2006b). The upper levels would have a smaller number of clusters which represent a more global structure. After several levels, all the verbs are expected to be grouped into one cluster. The number of levels and clusters at each level can thus be learned automatically.

We therefore propose a method that uses the decaying similarity function to learn the hierarchical clustering structure. One simple modification to algorithm 1 is to set the number of clusters at level l

(m_l) to be $m_{l-1} - 1$. m is denoted as the number of clusters that have at least one member according to eq. 5. We start by treating each verb as a cluster at the bottom level. The algorithm stops when all the data points are merged into one cluster. The increasingly decaying similarity causes many clusters to have 0 members especially at lower levels, which are pruned in the tree extraction.

3.2.4 Adding constraints to HGFC

The basic version of HGFC makes no prior assumptions about the classification. It is useful for learning novel verb classifications from scratch. However, when wishing to extend an existing classification (e.g. VerbNet) it may be desirable to guide the clustering performance on the basis of information that is already known. We propose a constrained version of HGFC which makes uses of labels at the bottom level to learn upper level classifications. We do this by adding soft constraints to clustering, following Vlachos et al. (2009).

We modify the similarity matrix W as follows: If two verbs have different labels ($l_i \neq l_j$), the similarity between them is decreased by a factor a, and $a < 1$. We set a to 0.5 in the experiments. The resulting tree is generally consistent with the original classification. The influence of the underlying data (domain or features) is reduced according to a.

4 Experimental evaluation

We applied the clustering methods introduced in section 3 to the test sets described in section 2 and evaluated them both quantitatively and qualitatively, as described in the subsequent sections.

4.1 Evaluation methods

We used class based accuracy (ACC) and adjusted rand index (R_{adj}) to evaluate the results on the flat test set T1 (see section 2 for details of T1-T3).

ACC is the proportion of members of dominant clusters DOM-CLUST$_i$ within all classes c_i.

$$\mathrm{ACC} = \frac{\sum_{i=1}^{C} \text{verbs in DOM-CLUST}_i}{\text{number of verbs}}$$

The formula of R_{adj} is (Hubert and Arabie, 1985):

$$\mathrm{R}_{adj} = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}] - \sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}/\binom{n}{2}}$$

where n_{ij} is the size of the intersection between class i and cluster j.

We used normalized mutual information (NMI) and F-Score (F) to evaluate hierarchical clustering results on T2 and T3. NMI measures the amount of statistical information shared by two random variables representing the clustering result and the gold-standard labels. Given random variables A and B:

$$\mathrm{NMI}(A,B) = \frac{I(A;B)}{[H(A)+H(B)]/2}$$

$$\mathrm{I}(A,B) = \sum_k \sum_j \frac{|(v_k \cap c_j|}{N} \log \frac{N|v_k \cap c_j|}{|v_k||c_j|}$$

where $|v_k \cap c_j|$ is the number of shared membership between cluster v_k and gold-standard class c_j. The normalized variant of mutual information (MI) enables the comparison of clustering with different cluster numbers (Manning et al., 2008).

F is the harmonic mean of precision (P) and recall (R). P is calculated using modified purity – a global measure which evaluates the mean precision of clusters. Each cluster is associated with its prevalent class. The number of verbs in a cluster K that take this class is denoted by $n_{prevalent}$(K).

$$\mathrm{mPUR} = \frac{\sum_{n_{prevalent(k_i)>2}} n_{prevalent(k_i)}}{\text{number of verbs}}$$

R is calculated using ACC.

$$\mathrm{F} = \frac{2 \cdot \mathrm{mPUR} \cdot \mathrm{ACC}}{\mathrm{mPUR} + \mathrm{ACC}}$$

F is not suitable for comparing results with different cluster numbers (Rosenberg and Hirschberg, 2007). Therefore, we only report NMI when the number of classes in clustering and gold-standard is substantially different.

Finally, we supplemented quantitative evaluation with qualitative evaluation of clusters produced by different methods.

4.2 Quantitative evaluation

We first evaluated AGG and the basic (unconstrained) HGFC on the small flat test set T1. The main purpose of this evaluation was to compare the results of our methods against previously published results on the same test set. The number of clusters (K) and levels (L) were inferred automatically for HGFC as described in section 3.2.3. However, to

make the results comparable with previously published ones, we cut the resulting hierarchy at the level of closest match (12 clusters) to the K (13) in the gold-standard. For AGG, we cut the hierarchy at 13 clusters.

Method	ACC	R_{adj}
HGFC	41.2	17.4
AGG (reproduced)	32.7	9.9
AGG (Stevenson and Joanis (2003)	31.0	9.0

Table 1: Comparison against Stevenson and Joanis (2003)'s result on T1 (using similar features).

Table 1 shows our results and the results of Stevenson and Joanis (2003) on T1 when employing AGG using Ward as the linkage criterion. In this experiment, we used the same feature set as Stevenson and Joanis (2003) (set B, see section 3.1) and were therefore able to reproduce their AGG result with a difference smaller than 2%. When using this simple feature set, HGFC outperforms the best performing AGG clearly: 8.5% in ACC and 7.3% in R_{adj}.

We also compared HGFC against the best reported clustering method on T1 to date – that of spectral clustering by Sun and Korhonen (2009). We used the feature sets C and D which are similar to the features (SCF parameterized by lexical prefences) in their experiments. HGFC obtains F of 49.93% on T1 which is 5% lower than the result of Sun and Korhonen (2009). The difference comes from the tree consistency requirement. When the HGFC is forced to produce a flat clustering (a one level tree only), it achieves the F of 52.55% which is very close to the performance of spectral clustering.

We then evaluated our methods on the hierarchical test sets T2 and T3. In the first set of experiments, we pre-defined the tree structure for HGFC by setting L to 3 and K at each level to be the K in the hierarchical gold standard. The hierarchy produced by AGG was cut into 3 levels according to Ks in the gold standard. This enabled direct evaluation of the results against the 3 level gold standards using both NMI and F.

The results are reported in tables 2 and 3. In these tables, N_c is the number of clusters in HGFC clustering while N_l is the number of classes in the gold standard (the two do not always correspond perfectly because a few clusters have zero members).

N_c	N_l	HGFC unconstrained		AGG	
		NMI	F	NMI	F
130	133	57.31	36.65	54.22	32.62
114	117	54.67	37.96	51.35	32.44
50	51	37.75	40.00	32.61	32.78

Table 2: Performance on T2 using a pre-defined tree structure.

N_c	N_l	HGFC unconstrained		HGFC constrained		AGG	
		NMI	F	NMI	F	NMI	F
31	32	51.65	42.01	91.47	92.07	49.70	40.30
15	14	42.75	47.70	82.16	82.80	39.19	43.69
11	11	38.91	51.17	71.69	75.00	34.88	44.80

Table 3: Performance on T3 using a pre-defined tree structure.

Table 2 compares the results of the unconstrained version of HGFC against those of AGG on our largest test set T2. As with T1, HGFC outperforms AGG clearly. The benefit can now be seen at 3 different levels of hierarchy. On average, the HGFC outperforms AGG 3.5% in NMI and 4.8% in F. The difference between the methods becomes clearer when moving towards the upper levels of the hierarchy.

Table 3 shows the results of both unconstrained and constrained versions of HGFC and those of AGG on the test set T3 (where singular classes are removed to enable proper evaluation of the constrained method). The results are generally generally better on this test set than on T2 – which is to be expected since T3 is a refined subset of T2[1].

Recall that the constrained version of HGFC learns the upper levels of classification on the basis of soft constraints set at the bottom level, as described earlier in section 3.2.4. As a consequence, NMI and F are both greater than 90% at the bottom level and the results at the top level are notably lower because the impact of the constraints degrades the further away one moves from the bottom level. Yet, the relatively high result across all levels shows that the constrained version of HGFC can be employed a useful method to extend the hierarchical structure of known classifications.

[1] NMI is higher on T2, however, because NMI has a higher baseline for larger number of clusters (Vinh et al., 2009). NMI is not ideal for comparing the results of T2 and T3.

T2			T3		
N_c	N_l	HGFC	N_c	N_l	HGFC
148	133	53.26	64	32	54.91
97	117	49.85	35	32	50.83
46	51	33.55	20	14	44.02
19	51	25.80	10	14	34.41
9	51	19.17	6	11	32.27
3	51	13.06			

Table 4: NMI of unconstrained HGFC when trees for T2 and T3 are inferred automatically.

Finally, Table 4 shows the results for the unconstrained HGFC on T2 and and T3 when the tree structure is not pre-defined but inferred automatically as described in section 3.2.3. 6 levels are learned for T2 and 5 for T3. The number of clusters produced ranges from 3 to 148 for T2 and from 6 to 64 for T3. We can see that the automatically detected cluster numbers distribute evenly across different levels. The scale of the clustering structure is more complete here than in the gold standards.

In the table, N_c indicates the number of clusters in the inferred tree, while N_l indicates the closest match to the number of classes in the gold standard. This evaluation is not fully reliable because the match between the gold standard and the clustering is poor at some levels of hierarchy. However, it is encouraging to see that the results do not drop dramatically until the match between the two is really poor.

4.3 Qualitative evaluation

To gain a better insight into the performance of HGFC, we conducted further qualitative analysis of the clusters the two versions of this method produced for T3. We focussed on the top level of 11 clusters (in the evaluation against the hierarchical gold standard, see table 3) as the impact of soft constraints is the weakest for the constrained method at this level.

As expected, the constrained HGFC kept many individual verbs belonging to same Verbnet subclass together (e.g. verbs *enjoy, hate, disdain, regret, love, despise, detest, dislike, fear* for the class 31.2.1) so that most clusters simply group lower level classes and their members together. Three nearly clean clusters were produced which only include sub-classes of the same class (e.g. 31.2.0 and 31.2.1 which both belong to 31.2 *Admire* verbs). However, the remaining 8 clusters group together sub-classes (and their members) belonging to unrelated parent classes. Interestingly, 6 of these make both syntactic and semantic sense. For example, several such 37.7 *Say* verbs and 29.5 *Conjencture* verbs are found together which share the meaning of communication and which take similar sentential complements.

In contrast, none of the clusters produced by the unconstrained HGFC represent a single VerbNet class. The majority represent a high number of classes and fewer members per class. Yet many of the clusters make syntactic and semantic sense. A good example is a cluster which includes member verbs from 9.7 *Spray/Load* verbs, 21.2 *Carve* verbs, 51.3.1 *Roll* verbs, and 10.4 *Wipe* verbs. The verbs included in this cluster share the meaning of specific type of motion and show similar syntactic behaviour.

Thorough Levin style investigation of especially the unconstrained method would require looking at shared diathesis alternations between cluster members. We left this for future work. However, the analysis we conducted confirmed that the constrained method could indeed be used for extending known classifications, while the unconstrained method is more suitable for acquiring novel classifications from scratch. The errors in clusters produced by both methods were mostly due to syntactic idiosyncracy and the lack of semantic information in clustering. We plan to address the latter problem in our future work.

5 Discussion and conclusion

We have introduced a new graph-based method – HGFC – to hierarchical verb clustering which avoids some of the problems (e.g. error propagation, pairwise cluster merging) reported with the frequently used AGG method. We modified HGFC so that it can be used to automatically determine the tree structure for clustering, and proposed two extensions to it which make it even more suitable for our task. The first involves automatically determining the number of clusters to be produced, which is useful when this is not known in advance. The second involves adding soft constraints to guide the clustering performance, which is useful when aiming to extend existing classification.

The results reported in the previous section are promising. On a flat test set (T1), the unconstrained version of HGFC outperforms AGG and performs very similarly with the best current flat clustering method (spectral clustering) evaluated on the same dataset. On the hierarchical test sets (T2 and T3), the unconstrained and constrained versions of HGFC outperform AGG clearly at all levels of classification. The constrained version of HGFC detects the missing hierarchy from the existing gold standards with high accuracy. When the number of clusters and levels is learned automatically, the unconstrained method produces a multi-level hierarchy. Our evaluation against a 3-level gold standard shows that such a hierarchy is fairly accurate. Finally, the results from our qualitative evaluation show that both constrained and unconstrained versions of HGFC are capable of learning valuable novel information not included in the gold standards.

The previous work on Levin style verb classification has mostly focussed on flat classifications using methods suitable for flat clustering (Schulte im Walde, 2006; Joanis et al., 2008; Sun et al., 2008; Li and Brew, 2008; Korhonen et al., 2008; Ó Séaghdha and Copestake, 2008; Vlachos et al., 2009). However, some works have employed hierarchical clustering as a method to infer flat clustering.

For example, Schulte im Walde and Brew (2002) employed AGG to initialize the KMeans clustering for German verbs. This gave better results than random initialization. Stevenson and Joanis (2003) used AGG for flat clustering on T1. They cut the hierarchy at the number of classes in the gold standard and found that it is difficult to automatically determine a good cut-off. Our evaluation in the previous section shows that HGFC outperforms their implementation of AGG.

AGG was also used by Ferrer (2004) who performed hierarchical clustering of 514 Spanish verbs. The results were evaluated against a hierarchical gold standard resembling that of Levin's classification in English (Vázquez et al., 2000). R_{adj} of 0.07 was reported for a 15-way classification which is comparable to the result of Stevenson and Joanis (2003).

Hierarchical clustering has also been performed for the related task of semantic verb classification. For example, Basili et al. (1993) identified the problems of AGG, and applied a conceptual clustering algorithm (Fisher, 1987) to Italian verbs. They used semi-automatically acquired semantic roles and the concept types as features. No quantitative results were reported. The qualitative evaluation shows that the resulting clusters are very fine-grained.

Schulte im Walde (2008) performed hierarchical clustering of German verbs using human verb association as features and AGG as a method. They focussed on two small collections of 56 and 104 verbs and evaluated the result against flat gold standard extracted from GermaNet (Kunze and Lemnitzer, 2002) and German FrameNet (Erk et al., 2003), respectively. They reported F of 62.69% for the 56 verbs, and F of 34.68% for the 104 verbs.

In the future, we plan to extend this research line in several directions. First, we will try to determine optimal features for different levels of clustering. For example, the general syntactic features (e.g. SCF) may perform the best at top levels of a hierarchy while more specific or refined features (e.g. SCF+pp) may be optimal at lower levels. We also plan to investigate incorporating semantic features, like verb selectional preferences, in our feature set. It is likely that different levels of clustering require more or less specific selectional preferences. One way to obtain the latter is hierarchical clustering of relevant noun data.

In addition, we plan to apply the unconstrained HGFC to specific domains to investigate its capability to learn novel, previously unknown classifications. As for the constrained version of HGFC, we will conduct a larger scale experiment on the VerbNet data to investigate what kind of upper level hierarchy it can propose for this resource (which currently has over 100 top level classes).

Finally, we plan to compare HGFC to other hierarchical clustering methods that are relatively new to NLP but have proved promising in other fields, including Bayesian Hierarchical Clustering (Heller and Ghahramani, 2005; Teh et al., 2008) and the method of Azran and Ghahramani (2006a) based on spectral clustering.

6 Acknowledgement

Our work was funded by the Royal Society University Research Fellowship (AK), the Dorothy Hodgkin Postgraduate Award (LS), the EPSRC

grants EP/F030061/1 and EP/G051070/1 (UK) and the EU FP7 project 'PANACEA'.

References

Arik Azran and Zoubin Ghahramani. A new approach to data driven clustering. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 57–64, New York, NY, USA, 2006a. ISBN 1-59593-383-2.

Arik Azran and Zoubin Ghahramani. Spectral methods for automatic multiscale data clustering. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pages 190–197. IEEE Computer Society Washington, DC, USA, 2006b.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *In COLING-ACL*, pages 86–90, 1998.

Roberto. Basili, Maria Teresa Pazienza, and Paola Velardi. Hierarchical clustering of verbs. In *Proceedings of the Workshop on Acquisition of Lexical Knowledge from Text*, 1993.

Nikoletta Bassiou and Constantine Kotropoulos. Long distance bigram models applied to word clustering. *Pattern Recogn.*, 44:145–158, January 2011. ISSN 0031-3203.

Ted Briscoe, John Carroll, and Rebecca Watson. The second release of the rasp system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, 2006.

Hoa Trang Dang. *Investigations into the Role of Lexical Semantics in Word Sense Disambiguation*. PhD thesis, CIS, University of Pennsylvania, 2004.

Katrin Erk, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. Towards a resource for lexical semantics: a large german corpus with extensive semantic annotation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 537–544, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Eva Esteve Ferrer. Towards a semantic classification of spanish verbs based on subcategorisation information. In *Proceedings of the ACL 2004 workshop on Student research*, ACLstudent '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987. ISSN 0885-6125.

David Graff. North american news text corpus. *Linguistic Data Consortium*, 1995.

Ralph Grishman, Catherine Macleod, and Adam Meyers. Comlex syntax: Building a computational lexicon. In *COLING*, pages 268–272, 1994.

Katherine A. Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pages 297–304. ACM, 2005. ISBN 1595931805.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985. ISSN 0176-4268.

Eric Joanis, Suzanne Stevenson, and David James. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367, 2008.

Karin Kipper. *VerbNet: A broad-coverage, comprehensive verb lexicon*. 2005.

Anna Korhonen, Yuval Krymolowski, and Nigel Collier. The Choice of Features for Classification of Verbs in Biomedical Texts. In *Proceedings of COLING*, 2008.

Claudia Kunze and Lothar Lemnitzer. GermaNet-representation, visualization, application. In *Proceedings of LREC*, 2002.

Geoffrey Leech. 100 million words of english: the british national corpus. *Language Research*, 28(1):1–13, 1992.

Beth. Levin. English verb classes and alternations: A preliminary investigation. *Chicago, IL*, 1993.

Jianguo Li and Chris Brew. Which Are the Best Features for Automatic Verb Classification. In *Proceedings of ACL*, 2008.

Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proceeding of the 17th international conference on World Wide Web*, pages 685–694, New York, NY, USA, 2008. ACM.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Yutaka Matsuo, Takeshi Sakaki, Kôki Uchiyama, and Mitsuru Ishizuka. Graph-based word clustering using a web search engine. In *Proceedings of the EMNLP*, pages 542–550, 2006.

George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

Travis E. Oliphant. Python for scientific computing. *Computing in Science and Engineering*, 9:10–20, 2007. ISSN 1521-9615.

Diarmuid Ó Séaghdha and Ann Copestake. Semantic classification with distributional kernels. In *Proceedings of COLING*, 2008.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

Judita Preiss, Ted Briscoe, and Anna Korhonen. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of ACL*, pages 912–919, 2007.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.

Sabine Schulte im Walde. Experiments on the automatic induction of german semantic verb classes. *Computational Linguistics*, 32(2), 2006.

Sabine Schulte im Walde. Human associations and the choice of features for semantic verb classification. *Research on Language and Computation*, 6:79–111, 2008. ISSN 1570-7075.

Sabine Schulte im Walde and Chris Brew. Inducing german semantic verb classes from purely syntactic subcategorisation information. In *Proceedings of ACL*, pages 223–230, 2002.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Lei Shi and Rada Mihalcea. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proceedings of CICLING*, 2005.

Suzanne Stevenson and Eric Joanis. Semi-supervised verb class discovery using noisy features. In *Proceedings of HLT-NAACL 2003*, pages 71–78, 2003.

Lin Sun and Anna Korhonen. Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of the EMNLP 2009*, 2009.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. Verb class discovery from rich syntactic data. *Lecture Notes in Computer Science*, 4919:16, 2008.

Robert Swier and Suzanne Stevenson. Unsupervised semantic role labelling. In *Proceedings of EMNLP*, pages 95–102, 2004.

Yee Whye Teh, Hal Daumé III, and Daniel Roy. Bayesian agglomerative clustering with coalescents. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

Akira Ushioda. Hierarchical clustering of words. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1159–1162. Association for Computational Linguistics, 1996.

Gloria Vázquez, Ana Fernández-Montraveta, and M. Antònia Martí. *Clasificación verbal:(alternancias de diátesis)*. Universitat de Lleida, 2000. ISBN 8484090671.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1073–1080, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. Unsupervised and constrained dirichlet process mixture models for verb clustering. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 74–82, 2009.

Joe H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963. ISSN 0162-1459.

Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, pages 1101–1113, 1993. ISSN 0162-8828.

Kai Yu, Shipeng Yu, and Volker Tresp. Soft clustering on graphs. *Advances in Neural Information Processing Systems*, 18:1553, 2006.

Beñat Zapirain, Eneko Agirre, and Lluís Màrquez. Robustness and generalization of role sets: PropBank vs. VerbNet. In *Proceedings of ACL-08: HLT*, pages 550–558, 2008.

Structured Lexical Similarity via Convolution Kernels on Dependency Trees

Danilo Croce
DII
University of Tor Vergata
00133 Roma, Italy
croce@info.uniroma2.it

Alessandro Moschitti
DISI
University of Trento
38123 Povo (TN), Italy
moschitti@disi.unitn.it

Roberto Basili
DII
University of Tor Vergata
00133 Roma, Italy
basili@info.uniroma2.it

Abstract

A central topic in natural language processing is the design of lexical and syntactic features suitable for the target application. In this paper, we study convolution dependency tree kernels for automatic engineering of syntactic and semantic patterns exploiting lexical similarities. We define efficient and powerful kernels for measuring the similarity between dependency structures, whose surface forms of the lexical nodes are in part or completely different. The experiments with such kernels for question classification show an unprecedented results, e.g. 41% of error reduction of the former state-of-the-art. Additionally, semantic role classification confirms the benefit of semantic smoothing for dependency kernels.

1 Introduction

A central topic in Natural Language Processing is the design of lexical and syntactic features suitable for the target application. The selection of effective patterns composed of syntactic dependencies and lexical constraints is typically a complex task.

Additionally, the availability of training data is usually scarce. This requires the development of generalized features or the definition of semantic similarities between them, e.g. as proposed in (Resnik, 1995; Jiang and Conrath, 1997; Schtze, 1998; Pedersen et al., 2004a; Bloehdorn and Moschitti, 2007b; Davis et al., 2007) or in semi-supervised settings, e.g. (Chapelle et al., 2006). A semantic similarity can be defined at structural level over a graph, e.g. (Freeman, 1977; Bunke and Shearer, 1998; Brandes, 2001; Zhao et al., 2009), as well as combining structural and lexical similarity over semantic networks, e.g. (Cowie et al., 1992; Wu and Palmer, 1994; Resnik, 1995; Jiang and Conrath, 1997; Schtze, 1998; Leacock and Chodorow, 1998; Pedersen et al., 2004a; Budanitsky and Hirst, 2006). More recent research also focuses on mechanisms to define if two structures, e.g. graphs, are enough similar, as explored in (Mihalcea, 2005; Zhao et al., 2009; Fürstenau and Lapata, 2009; Navigli and Lapata, 2010).

On one hand, previous work shows that there is a substantial lack of automatic methods for engineering lexical/syntactic features (or more in general syntactic/semantic similarity). On the other hand, automatic feature engineering of syntactic or shallow semantic structures has been carried out by means of structural kernels, e.g. (Collins and Duffy, 2002; Kudo and Matsumoto, 2003; Cumby and Roth, 2003; Cancedda et al., 2003; Daumé III and Marcu, 2004; Toutanova et al., 2004; Shen et al., 2003; Gliozzo et al., 2005; Kudo et al., 2005; Titov and Henderson, 2006; Zelenko et al., 2002; Bunescu and Mooney, 2005; Zhang et al., 2006). The main idea of structural kernels is to generate structures that in turn represent syntactic or shallow semantic features. Most notably, the work in (Bloehdorn and Moschitti, 2007b) encodes lexical similarity in such kernels. This is essentially the syntactic tree kernel (STK) proposed in (Collins and Duffy, 2002) in which syntactic fragments from constituency trees can be matched even if they only differ in the leaf nodes (i.e. they have different surface forms). This implies matching scores lower than 1, depending on the semantic similarity of the corresponding leaves in the syntactic fragments.

Although this kernel achieves state-of-the-art performance in NLP tasks, such as Question Classifica-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1034–1046,
Edinburgh, Scotland, UK, July 27–31, 2011.

tion (Bloehdorn and Moschitti, 2007b) and Textual Entailment (Mehdad et al., 2010), it offers clearly possibility of improvement: (i) better possibility to exploit semantic smoothing since, e.g., trivially STK only matches the syntactic structure *apple/orange* when comparing *the big beautiful apple* to *a nice large orange*; and (ii) STK cannot be effectively applied to dependency structures, e.g. see experiments and motivation in (Moschitti, 2006a). Additionally, to our knowledge, there is no previous study that clearly describes how dependency structures should be converted in trees to be fully and effectively exploitable by convolution kernels. Indeed, although the work in (Culotta and Sorensen, 2004) defines a dependency tree also using node similarity, it is not a convolution kernel: this results in a much poorer feature space.

In this paper, we propose a study of convolution kernels for dependency structures aiming at jointly modeling syntactic and lexical semantic similarity. More precisely, we define several dependency trees exploitable by the Partial Tree Kernel (PTK) (Moschitti, 2006a) and compared them with STK over constituency trees. Most importantly, we define an innovative and efficient class of kernels, i.e. the Smoothed Partial Tree Kernels (SPTKs), which can measure the similarity of structural similar trees whose nodes are associated with different but related lexicals. Given the convolution nature of such kernels any possible node path of lexicals provide a contribution smoothed by the similarity accounted by its nodes.

The extensive experimentation on two datasets of question classification (QC) and semantic role labeling (SRL), shows that: (i) PTK applied to our dependency trees outperforms STK, demonstrating that dependency parsers are fully exploitable for feature engineering based on structural kernels; (ii) SPTK outperforms any previous kernels achieving an unprecedented result of 41% of error reduction with respect to the former state-of-the-art on QC; and (iii) the experiments on SRL confirm that the approach can be applied to different tasks without any tuning and again achieving state-of-the-art accuracy.

In the reminder of this paper, Section 2 provides the background for structural and lexical similarity kernels. Section 3 introduces SPTK. Section 4 provides our representation models for dependency trees. Section 5 presents the experimental evaluation for QC and SRL. Section 6 derives the conclusions.

2 Kernel Background

In kernel-based machines, both learning and classification algorithms only depend on the inner product between instances. This in several cases can be efficiently and implicitly computed by kernel functions by exploiting the following dual formulation: $\sum_{i=1..l} y_i \alpha_i \phi(o_i)\phi(o) + b = 0$, where o_i and o are two objects, ϕ is a mapping from the objects to feature vectors $\vec{x_i}$ and $\phi(o_i)\phi(o) = K(o_i, o)$ is a kernel function implicitly defining such mapping. In case of structural kernels, K determines the shape of the substructures describing the objects above. The most general kind of kernels used in NLP are string kernels, e.g. (Shawe-Taylor and Cristianini, 2004), the Syntactic Tree Kernels (Collins and Duffy, 2002) and the Partial Tree Kernels (Moschitti, 2006a).

2.1 String Kernels

The String Kernels (SK) that we consider count the number of subsequences shared by two strings of symbols, s_1 and s_2. Some symbols during the matching process can be skipped. This modifies the weight associated with the target substrings as shown by the following SK equation:

$$SK(s_1, s_2) = \sum_{u \in \Sigma^*} \phi_u(s_1) \cdot \phi_u(s_2) = \sum_{u \in \Sigma^*} \sum_{\vec{I}_1 : u = s_1[\vec{I}_1]} \sum_{\vec{I}_2 : u = s_2[\vec{I}_2]} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)}$$

where, $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$ is the set of all strings, $\vec{I}_1$ and $\vec{I}_2$ are two sequences of indexes $\vec{I} = (i_1, ..., i_{|u|})$, with $1 \leq i_1 < ... < i_{|u|} \leq |s|$, such that $u = s_{i_1}..s_{i_{|u|}}$, $d(\vec{I}) = i_{|u|} - i_1 + 1$ (distance between the first and last character) and $\lambda \in [0, 1]$ is a decay factor.

It is worth noting that: (a) longer subsequences receive lower weights; (b) some characters can be omitted, i.e. gaps; (c) gaps determine a weight since the exponent of λ is the number of characters and gaps between the first and last character; and (c) the complexity of the SK computation is $O(mnp)$ (Shawe-Taylor and Cristianini, 2004), where m and n are the lengths of the two strings, respectively and p is the length of the largest subsequence we want to consider.

2.2 Tree Kernels

Convolution Tree Kernels compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. For this purpose, let the set $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$ be a tree fragment space and $\chi_i(n)$ be an indicator function, equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over T_1 and T_2 is $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \chi_i(n_1)\chi_i(n_2)$. The latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes. The Δ function determines the richness of the kernel space and thus different tree kernels. Hereafter, we consider the equation to evaluate STK and PTK [1].

2.2.1 Syntactic Tree Kernels (STK)

To compute STK is enough to compute $\Delta_{STK}(n_1, n_2)$ as follows (recalling that since it is a syntactic tree kernels, each node can be associated with a production rule): (i) if the productions at n_1 and n_2 are different then $\Delta_{STK}(n_1, n_2) = 0$; (ii) if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children then $\Delta_{STK}(n_1, n_2) = \lambda$; and (iii) if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then $\Delta_{STK}(n_1, n_2) = \lambda \prod_{j=1}^{l(n_1)} (1 + \Delta_{STK}(c_{n_1}^j, c_{n_2}^j))$, where $l(n_1)$ is the number of children of n_1 and c_n^j is the j-th child of the node n. Note that, since the productions are the same, $l(n_1) = l(n_2)$ and the computational complexity of STK is $O(|N_{T_1}||N_{T_2}|)$ but the average running time tends to be linear, i.e. $O(|N_{T_1}| + |N_{T_2}|)$, for natural language syntactic trees (Moschitti, 2006a).

2.2.2 The Partial Tree Kernel (PTK)

The computation of PTK is carried out by the following Δ_{PTK} function: if the labels of n_1 and n_2 are different then $\Delta_{PTK}(n_1, n_2) = 0$; else $\Delta_{PTK}(n_1, n_2) =$

$$\mu\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1) = l(\vec{I}_2)} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_{PTK}(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big)$$

where $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11} + 1$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21} + 1$. This way, we penalize both larger trees and child subsequences with gaps. PTK is more general than the STK as if we only consider the contribution of shared subsequences containing all children of nodes, we implement the STK kernel. The computational complexity of PTK is $O(p\rho^2|N_{T_1}||N_{T_2}|)$ (Moschitti, 2006a), where p is the largest subsequence of children that we want consider and ρ is the maximal outdegree observed in the two trees. However the average running time again tends to be linear for natural language syntactic trees (Moschitti, 2006a).

2.3 Lexical Semantic Kernel

Given two text fragments d_1 and $d_2 \in D$ (the text fragment set), a general lexical kernel (Basili et al., 2005) defines their similarity as:

$$K(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} (\omega_1 \omega_2) \times \sigma(w_1, w_2) \quad (1)$$

where ω_1 and ω_2 are the weights of the words (features) w_1 and w_2 in the documents d_1 and d_2, respectively, and σ is a term similarity function, e.g. (Pedersen et al., 2004b; Sahlgren, 2006; Corley and Mihalcea, 2005; Mihalcea et al., 2005). Technically, any σ can be used, provided that the resulting Gram matrix, $\boldsymbol{G} = K(d_1, d_2)\ \forall d_1, d_2 \in D$ is positive semi-definite (Shawe-Taylor and Cristianini, 2004) (D is typically the training text set).

We determine the term similarity function through distributional analysis (Pado and Lapata, 2007), according to the idea that the meaning of a word can be described by the set of textual contexts in which it appears (*Distributional Hypothesis*, (Harris, 1964)). The contexts are words appearing in a n-window with target words: such a space models a generic notion of semantic relatedness, i.e. two words close in the space are likely to be either in paradigmatic or syntagmatic relation as in (Sahlgren, 2006). The original word-by-word context matrix M is decomposed through Singular Value Decomposition (SVD) (Golub and Kahan, 1965) into the product of three new matrices: U, S, and V so that S is diagonal and $M = USV^T$. M is approximated by $M_l = U_l S_l V_l^T$ in which only the first l columns of U and V are used, and only the first l greatest singular values are considered. This approximation supplies a way to project a generic term w_i into the l-

[1] To have a similarity score between 0 and 1, a normalization in the kernel space, i.e. $\frac{TK(T_1,T_2)}{\sqrt{TK(T_1,T_1) \times TK(T_2,T_2)}}$ is applied.

dimensional space using $W = U_l S_l^{1/2}$, where each row corresponds to the representation vectors $\vec{w_i}$. Therefore, given two words w_1 and w_2, the term similarity function σ is estimated as the cosine similarity between the corresponding projections $\vec{w_1}, \vec{w_2}$, i.e $\sigma(w_1, w_2) = \frac{\vec{w_1} \cdot \vec{w_2}}{\|\vec{w_1}\| \|\vec{w_2}\|}$. The latent semantic kernels (Siolas and d'Alch Buc, 2000; Cristianini et al., 2001) derive G by applying LSA, resulting in a valid kernel.

Another methods to design a valid kernel is to represent words as word vectors and compute σ as their scalar product between such vectors. For example, in (Bloehdorn et al., 2006), bag of hyponyms and hypernyms (up to a certain level of WordNet hierarchy) were used to build such vectors. We will refer to such similarity as WL (word list).

3 Smoothing Partial Tree Kernel (SPTK)

Combining lexical and structural kernels provides clear advantages on all-vs-all words similarity, which tends to semantically diverge. Indeed syntax provides the necessary restrictions to compute an effective semantic similarity. Following this idea, Bloedhorn & Moschitti (2007a) modified step (i) of Δ_{STK} computation as follows: (i) if n_1 and n_2 are pre-terminal nodes with the same number of children, $\Delta_{STK}(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} \sigma(lex(n_1), lex(n_2))$, where lex returns the node label. This allows to match fragments having same structure but different leaves by assigning a score proportional to the product of the lexical similarities of each leaf pair. Although it is an interesting kernel, the fact that lexicals must belong to the leaf nodes of exactly the same structures limits its applications. Trivially, it cannot work on dependency trees. Hereafter, we define a much more general smoothed tree kernel that can be applied to any tree and exploit any combination of lexical similarities, respecting the syntax enforced by the tree.

3.1 SPTK Definition

If n_1 and n_2 are leaves then $\Delta_\sigma(n_1, n_2) = \mu\lambda\sigma(n_1, n_2)$; else

$$\Delta_\sigma(n_1, n_2) = \mu\sigma(n_1, n_2) \times \Big(\lambda^2 + \sum_{\vec{I_1}, \vec{I_2}, l(\vec{I_1}) = l(\vec{I_2})} \lambda^{d(\vec{I_1}) + d(\vec{I_2})} \prod_{j=1}^{l(\vec{I_1})} \Delta_\sigma(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big), \quad (2)$$

where σ is any similarity between nodes, e.g. between their lexical labels, and the other variables are the same of PTK.

3.2 Soundness

A completely formal proof of the validity of the Eq. 2 is beyond the purpose of this paper (mainly due to space reason). Here we give a first sketch: let us consider σ as a string matching between node labels and $\lambda = \mu = 1$. Each recursive step of Eq. 2 can be seen as a summation of $(1 + \prod_{j=1}^{l(\vec{I_1})} \Delta_{STK}(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j})))$, i.e. the Δ_{STK} recursive equation (see Sec. 2.2.1), for all subsequences of children $c_{n_1}(\vec{I}_{1j})$. In other words, PTK is a summation of an exponential number of STKs, which are valid kernels. It follows that PTK is a kernel. Note that the multiplication by λ and μ elevated to any power only depends on the target fragment. Thus, it just gives an additional weight to the fragment and does not violate the Mercer's conditions. In contrast, the multiplication by $\sigma(n_1, n_2)$ does depend on both comparing examples, i.e. on n_1 and n_2. However, if the matrix $\big[\sigma(n_1, n_2)\big] \forall n_1, n_2 \in f \in \mathcal{F}$ is positive semi-definite, a decomposition exists such that $\sigma(n_1, n_2) = \phi(n_1)\phi(n_2) \Rightarrow \Delta_\sigma(n_1, n_2)$ can be written as $\sum_{i=1}^{|\mathcal{F}|} \phi(n_1)\chi_i(n_1)\phi(n_2)\chi_i(n_2) = \sum_{i=1}^{|\mathcal{F}|} \phi_\sigma(n_1)\phi_\sigma(n_2)$ (see Section 2.2), which proves SPTK to be a valid kernel.

3.3 Efficient Evaluation

We followed the idea in (Moschitti, 2006a) for efficiently computing SPTK. We consider Eq. 2 evaluated with respect to sequences of different length p; it follows that

$$\Delta(n_1, n_2) = \mu\sigma(n_1, n_2)\big(\lambda^2 + \sum_{p=1}^{m} \Delta_p(c_{n_1}, c_{n_2})\big),$$

where Δ_p evaluates the number of common subtrees rooted in subsequences of exactly p children (of n_1 and n_2) and $m = min\{l(c_{n1}), l(c_{n2})\}$. Given the two child sequences $s_1 a = c_{n_1}$ and $s_2 b = c_{n_2}$ (a and b are the last children)

$$\Delta_p(s_1 a, s_2 b) = \Delta(a, b) \times \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1| - i + |s_2| - r} \times \Delta_{p-1}(s_1[1:i], s_2[1:r])$$

where $s_1[1:i]$ and $s_2[1:r]$ are the child subsequences from 1 to i and from 1 to r of s_1 and s_2. If we name the double summation term as D_p, we can

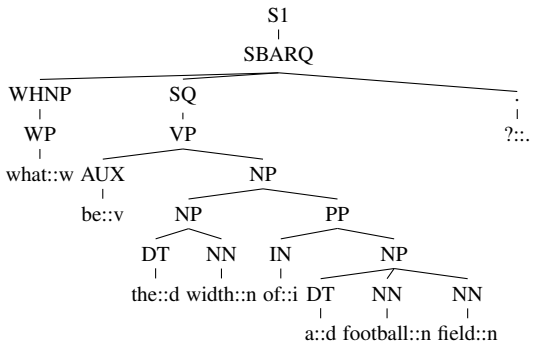


Figure 1: Constituent Tree (CT)

rewrite the relation as:

$$\Delta_p(s_1 a, s_2 b) = \begin{cases} \Delta(a,b) D_p(|s_1|,|s_2|) \text{ if } \sigma(a,b) > 0; \\ 0 \qquad\qquad\qquad otherwise. \end{cases}$$

Note that D_p satisfies the recursive relation:

$$D_p(k,l) = \Delta_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k,l-1) + \lambda D_p(k-1,l) - \lambda^2 D_p(k-1,l-1)$$

By means of the above relation, we can compute the child subsequences of two sequences s_1 and s_2 in $O(p|s_1||s_2|)$. Thus the worst case complexity of the SPTK is identical to PTK, i.e. $O(p\rho^2|N_{T_1}||N_{T_2}|)$, where ρ is the maximum branching factor of the two trees. The latter is very small in natural language parse trees and we also avoid the computation of node pairs with non similar labels.

We note that PTK generalizes both (i) SK, allowing the similarity between sequences (node children) structured in a tree and (ii) STK, allowing the computation of STK over any possible pair of subtrees extracted from the original tree. For this reason, we do not dedicate additional space on the definition of the smoothed SK or smoothed STK, which are in any case important corollary findings of our research.

3.4 Innovative Features of SPTK

The most similar kernel to SPTK is the Syntactic Semantic Tree Kernel (SSTK) proposed in (Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b). However, the following aspects show the remarkable innovativeness of SPTK:

- SSTK can only work on constituency trees and not on dependency trees (see (Moschitti, 2006a)).
- The lexical similarity in SSTK is only applied to leaf nodes in exactly the same syntactic constituents. Only complete matching of the structure of subtrees is allowed: there is absolutely no flexibility, e.g. the NP structure "cable television system" has no match with the NP "video streaming system". SPTK provides matches between all possible relevant subparts, e.g. "television system" and "video system" (so also exploiting the meaningful similarity between "video" and "television").
- The similarity in the PTK equation is added such that SPTK still corresponds to a scalar product in the semantic/structure space[2].
- We have provided a fast evaluation of SPTK with dynamic programming (otherwise the computation would have required exponential time).

4 Dependency Tree Structures

The feature space generated by the structural kernels, presented in the previous section, obviously depends on the input structures. In case of PTK and SPTK different tree representations may lead to engineer more or less effective syntactic/semantic feature spaces. The next two sections provide our representation models for dependency trees and their discussion.

4.1 Proposed Computational Structures

Given the following sentence:

(s1) *What is the width of a football field?*

The representation tree for a phrase structure paradigm leaves little room for variations as shown by the constituency tree (CT) in Figure 1. We apply lemmatization to the lexicals to improve generalization and, at the same time, we add a generalized PoS-tag, i.e. noun (n::), verb (v::), adjective (::a), determiner (::d) and so on, to them. This is useful to measure similarity between lexicals belonging to the same grammatical category.

In contrast, the conversion of dependency structures in computationally effective trees (for the above kernels) is not straightforward. We need to decide the role of lexicals, their grammatical functions (GR), PoS-tags and dependencies. It is natural

[2]This is not trivial: for example if sigma is added in Eq. 2 by only multiplying the λ^{d1+d2} term, no valid space is generated.

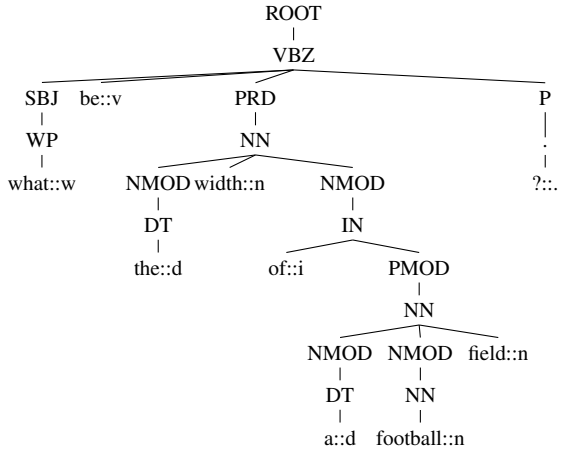


Figure 2: PoS-Tag Centered Tree (PCT)

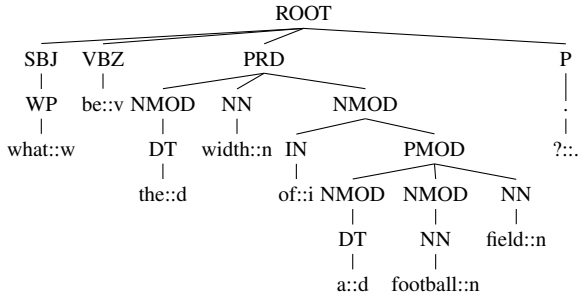


Figure 3: Grammatical Relation Centered Tree (GRCT)

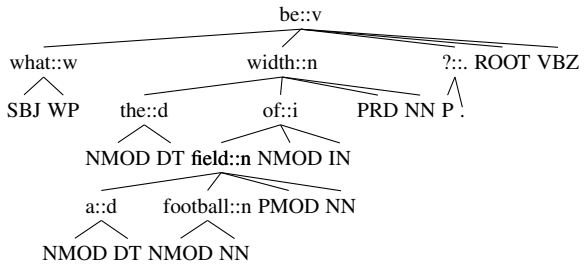


Figure 4: Lexical Centered Tree (LCT)

to associate edges with dependencies but, since our kernels cannot process labels on the arcs, they must be associated with tree nodes. The basic idea of our structures is to use (i) one of the three kinds of information above as central node, from which dependencies are drawn and (ii) all the other information as features (in terms of additional nodes) attached to the central nodes.

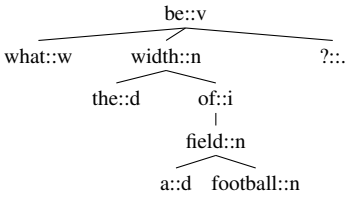


Figure 5: Lexical Only Centered Tree (LOCT)

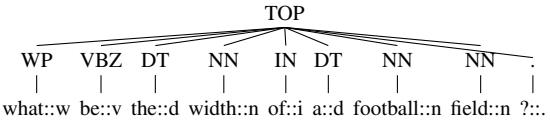


Figure 6: Lexical and PoS-Tag Sequences Tree (LPST)

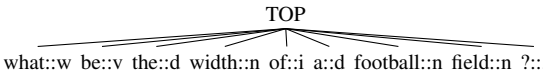


Figure 7: Lexical Sequences Tree (LST)

We define three main trees: the PoS-Tag Centered Tree (PCT), e.g. see Figure 2, where the GR is added as father and the lexical as a child; the GR Centered Tree (GRCT), e.g. see Figure 3, where the PoS-Tags are children of GR nodes and fathers of their associated lexicals; and the Lexical Centered Tree (LCT), e.g. see Figure 4, in which both GR and PoS-Tag are added as the rightmost children.

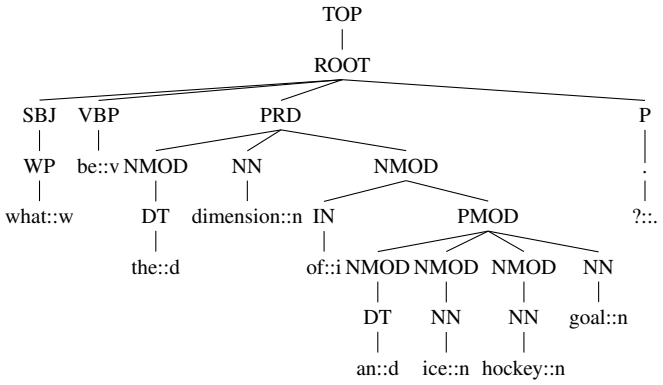


Figure 8: Grammatical Relation Centered Tree of (s2)

4.2 Comparative Structures

To better study the role of the above dependency structures, especially from a performance perspective, we define additional structures: the Lexical Only Centered Tree (LOCT), e.g. see Figure 5, which is an LCT only containing lexical nodes; the Lexical and PoS-Tag Sequences Tree (LPST), e.g. see Figure 6, which ignores the syntactic structure of the sentence being a simple sequence of PoS-Tag nodes, where lexicals are simply added as children; and the Lexical Sequence Tree (LST), where only lexical items are leaves of a single root node. PTK

and PSTK applied to it simulates a standard SK and an SK with smoothing, respectively.

4.3 Structural Features

Section 2 has already described the kind of features generated by SK, STK and PTK. However, it is interesting to analyze what happens when SPTK is applied. For example, given the following sentence syntactically and semantically similar to s1:

(s2) *What is the dimension of an ice hockey goal?*

Figure 8 shows the corresponding GRCT, whose largest PTK fragment shared with the GRTC of s1 (Fig. 3) is: *(ROOT (SBJ (WP (what::w))) (PRD (NMOD (DT (the::d))) (NN) (NMOD (IN (of::i)) (PMOD (NMOD (DT)) (NMOD (NN)) (NN)))) (P (. (?::.))))*. If smoothing is applied the matching is almost total, i.e. also the children: *width::n/dimension::n*, *football::n/hockey::n* and *field::n/goal::n* will be matched (with a smoothing equal to the product of their similarities).

The matching using LCT is very interesting: without smoothing, the largest subtree is: *(be::v (what::w (SBJ) (WP)) (ROOT))*; when smoothing is used only the fragment *(NMOD (NN (ice::n))* will not be part of the match. This suggests that LCT will probably receive the major benefit from smoothing. Additionally, with respect to all the above structures, LCT is the only one that can produce only lexical fragments, i.e. paths only composed by similar lexical nodes constrained by syntactic dependencies. All the other trees produce fragments in which lexicals play the role of features of GR or PoS-Tag nodes.

5 Experiments

The aim of the experiments is to analyze different levels of representation, i.e. structure, for syntactic dependency parses. At the same time, we compare with the constituency trees and different kernels to derive the best syntactic paradigm for convolution kernels. Most importantly, the role of lexical similarity embedded in syntactic structures will be investigated. For this purpose, we first carry out extensive experiments on coarse and fine grained QC and then we verify our findings on a completely different task, i.e. Argument Classification in SRL.

5.1 General experimental setup

Tools: for SVM learning, we extended the SVM-LightTK software[3] (Moschitti, 2006a) (which includes structural kernels in SVMLight (Joachims, 2000)) with the smooth match between tree nodes. For generating constituency trees, we used the Charniak parser (Charniak, 2000) whereas we applied LTH syntactic parser (described in (Johansson and Nugues, 2008a)) to generate dependency trees.

Lexical Similarity: we used the Eq. 1 with $\omega_1 = \omega_2 = 1$ and σ is derived with both approaches described in Sec. 2.3. The first approach is LSA-based: LSA was applied to ukWak (Baroni et al., 2009), which is a large scale document collection made by 2 billion tokens. More specifically, to build the matrix M, POS tagging is first applied to build rows with pairs $\langle$lemma, ::POS$\rangle$, or lemma::POS in brief. The contexts of such items are the columns of M and are short windows of size $[-3, +3]$, centered on the items. This allows for better capturing syntactic properties of words. The most frequent 20,000 items are selected along with their 20k contexts. The entries of M are the point-wise mutual information between them. The SVD reduction is then applied to M, with a dimensionality cut of $l = 250$. The second approach uses the similarity based on word list (WL) as provided in (Li and Roth, 2002).

Models: SVM-LightTK is applied to the different tree representations discussed in Section 4. Since PTK and SPTK are typically used in our experiments, to have a more compact acronym for each model, we associate the latter with the name of the structure, i.e. this indicates that PTK is applied to it. Then the presence of the subscript $_{WL}$ and $_{LSA}$ indicates that SPTK is applied along with the corresponding similarity, e.g. LCT$_{WL}$ is the SPTK kernel applied to LCT structure, using WL similarity. We experiment with multi-classification, which we model through *one-vs-all* scheme by selecting the category associated with the maximum SVM margin. The quality of such classification is measured with accuracy. We determine the statistical significance by using the model described in (Yeh, 2000) and implemented in (Padó, 2006).

The parameterization of each classifier is carried on a held-out set (30% of the training) and concerns with the setting of the trade-off parameter (option -c) and the Leaf Weight (LeW) (see Sec. 5.2), which is used to linearly scale the contribution of the leaf nodes. In contrast, the cost-factor parameter of the SVM-LightTK is set as the ratio between the num-

[3] http://disi.unitn.it/moschitti/Tree-Kernel.htm

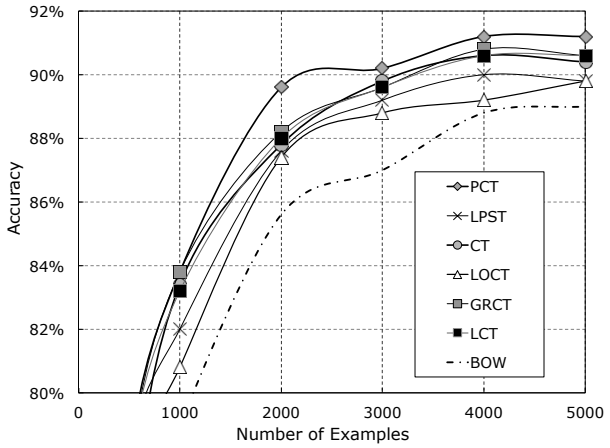


Figure 9: Learning curves: comparison with no similarity

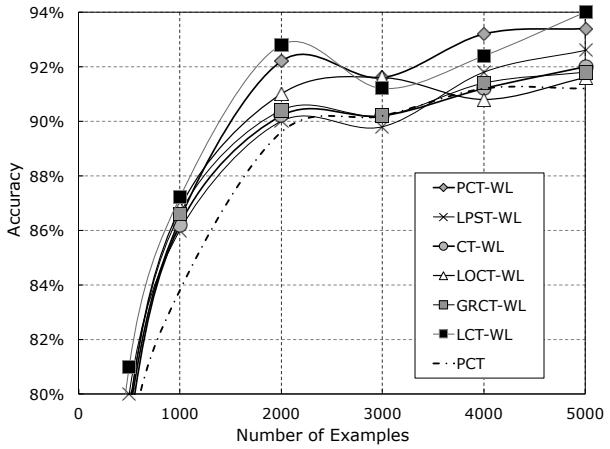


Figure 10: Learning curves: comparison with similarity

ber of negative and positive examples for attempting to have a balanced Precision/Recall.

5.2 QC experiments

For these experiments, we used the UIUC dataset (Li and Roth, 2002). It is composed by a training set of 5,452 questions and a test set of 500 questions[4]. Question classes are organized in two levels: 6 coarse-grained classes (like `ENTITY` or `HUMAN`) and 50 fine-grained sub-classes (e.g. `Plant`, `Food` as subclasses of `ENTITY`).

The outcome of the several kernels applied to several structures for the coarse and fine grained QC is reported in Table 1. The first column shows the experimented models, obtained by applying PTK/SPTK to the structures described in Sec. 4. The last two rows are: CT-STK, i.e. STK applied to a constituency tree and BOW, which is a linear kernel applied to lexical vectors. Column 2, 3 and 4 report the accuracy using no, LSA and WL similarity, where LeW is the amplifying parameter, i.e. a weight associated with the leaves in the tree. The last three columns refer to the fine grained task.

It is worth nothing that when no similarity is applied: (i) BOW produces high accuracy, i.e. 88.8% but it is improved by STK (the current state-of-the-art[5] in QC (Zhang and Lee, 2003; Moschitti et al., 2007)); (ii) PTK applied to the same tree of STK produces a slightly lower value (non-statistically significant difference); (iii) interestingly, when PTK is instead applied to dependency structures, it improves STK, i.e. 91.60% vs 91.40% (although not significantly); and (iv) LCT, strongly based on lexical nodes, is the least accurate, i.e 90.80% since it is obviously subject to data sparseness (fragments only composed by lexicals are very sparse).

The very important results can be noted when lexical similarity is used, i.e. SPTK is applied: (a) all the syntactic-base structures using both LSA or WL improve the classification accuracy. (b) CT gets the lowest improvement whereas LCT achieves an impressive result of 94.80%, i.e more than 41% of relative error reduction. It seems that the lexical similar paths when driven by syntax produces accurate features. Indeed, when syntax is missing such as for the unstructured lexical path of LST_{LSA}, the accuracy does not highly improve or may also decrease. Additionally, the result of our best model is so high that its errors only refer to questions like *What did Jesse Jackson organize ?*, where the classifier selected *Entity* instead of $Human$ category. These refer to clear cases where a huge amount of background knowledge is needed for deriving the exact solution.

Finally, on the fine grained experiments LCT still produces the most accurate outcome again exceeding the state-of-the-art (Zhang and Lee, 2003), where WL significantly improves on all models (CT included).

5.3 Learning curves

It is interesting to study the impact of syntactic/semantic kernels on the learning generalization. For this purpose, Fig. 9 reports the learning curve

[4] http://cogcomp.cs.illinois.edu/Data/QA/QC/

[5] Note that in (Bloehdorn and Moschitti, 2007b), higher accuracy values for smoothed STK are shown for different parameters but the best according to a validation set is not highlighted.

	COARSE						FINE					
	NO		LSA		WL		NO		LSA		WL	
	LeW	Acc.	LeW	Acc.	LeW	Acc.	LeW	Acc.	LeW	Acc.	LeW	Acc.
CT	4	90.80%	2	91.00%	5	92.20%	4	84.00%	5	83.00%	7	86.60%
GRCT	3	**91.60%**	4	92.60%	2	**94.20%**	3	83.80%	4	83.20%	2	85.00%
LCT	1	90.80%	1	**94.80%**	1	**94.20%**	0.33	**85.40%**	1	86.20%	0.33	**87.40%**
LOCT	1	89.20%	1	93.20%	1	91.80%	1	**85.40%**	1	**86.80%**	1	87.00%
LST	1	88.20%	1	85.80%	1	89.60%	1	84.00%	1	80.00%	1	85.00%
LPST	3	89.40%	1	89.60%	1	92.40%	3	84.20%	4	82.20%	1	84.60%
PCT	4	91.20%	4	92.20%	5	93.40%	4	84.80%	5	84.00%	5	85.20%
CT-STK	-	91.20%	-	-	-	-	-	82.20%	-	-	-	-
BOW	-	88.80%	-	-	-	-	-	83.20%	-	-	-	-

Table 1: Accuracy of structural several kernels on different structures for coarse and fine grained QC

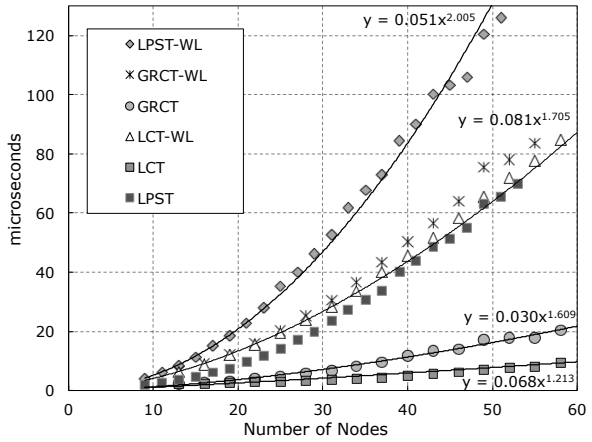


Figure 11: Micro-seconds for each kernel computation

of the previous models without lexical similarity whereas Fig. 10 shows the complete SPTK behavior through the different structures. We note that when no similarity is used the dependency trees better generalize than constituency trees or non-syntactic structures like LPST or BOW. When WL is activated, all models outperform the best kernel of the previous pool, i.e. PCT (see dashed line of Fig. 10 or the top curve in Fig. 9).

5.4 Kernel Efficiency

We plotted the average running time of each computation of PTK/SPTK applied to the different structures. We divided the examples from QC based on the number of nodes in each example. Figure 11 shows the elapsed time in function of the number of nodes for different tree representations. We note that: (i) when the WL is not active, LCT and GRCT are very fast as they impose hierarchical matching of subtrees; (ii) when the similarity is activated, LCT$_{WL}$ and GRCT$_{WL}$ tend to match many more tree fragments thus their complexity increases. However, the equations of the curve fit, shown in the figure, suggests that the trend is sub-quadratic ($x^{1.7}$). Only LPST$_{WL}$, which has no structure, matches a very large number of sequences of nodes, when the similarity is active. This increases the complexity, which results in an order higher than 2.

5.5 FrameNet Role Classification Experiments

To verify that our findings are general and that our syntactic/semantic dependency kernels can be effectively exploited for diverse NLP tasks, we experimented with a completely different application, i.e. FrameNet SRL classification (gold standard boundaries). We used the FrameNet version 1.3 with the 90/10% split between training and test set (i.e 271,560 and 30,173 examples respectively), as defined in (Johansson and Nugues, 2008b), one of the best system for FrameNet parsing. We used the LTH dependency parser. LSA was applied to the BNC corpus, the source of the FrameNet annotations.

For each of 648 frames, we applied SVM along with the best models for QC, i.e. GRCT and LCT, to learn its associated binary role classifiers (RC) for a total of 4,254 classifiers. For example, Figure 12 shows the LCT representation of the first two roles of the following sentence:

$[Bootleggers]_{\text{CREATOR}}, then$ **copy** $[the\ film]_{\text{ORIGINAL}}$ $[onto\ hundreds\ of\ VHS\ tapes]_{\text{GOAL}}$

Table 2 shows the results of the different multi-classifiers. GRCT and LCT show a large accuracy, i.e. 87.60. This improves up to 88.74 by activating the LSA similarity. The combination GRCT$_{LSA}$+LCT$_{LSA}$ significantly improves the above model, achieving 88.91%. This is very close to the state-of-the-art of SRL for classification (using a single classifier, i.e. no joint model), i.e. 89.6%, achieved in (Johansson and Nugues, 2008b).

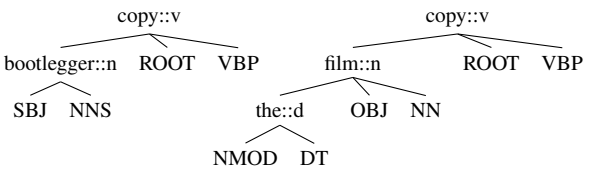


Figure 12: LCT Examples for argument roles

Kernel	Accuracy
GRCT	87.60%
$GRCT_{LSA}$	88,61%
LCT	87.61%
LCT_{LSA}	88.74%
GRCT + LCT	87.99%
$GRCT_{LSA}$ + LCT_{LSA}	**88.91%**

Table 2: Argument Classification Accuracy

Finally, it should be noted that, to learn and test the SELF_MOTION multi-classifier, containing 14,584 examples, distributed on 22 roles, SVM-SPTK employed 1.5 h and 10 minutes, respectively[6].

6 Final Remarks and Conclusion

In this paper, we have proposed a study on representation of dependency structures for the design of effective structural kernels. Most importantly, we have defined a new class of kernel functions, i.e. SPTKs, that carry out syntactic and lexical similarities on the above structures. SPTK exploits the latter by providing generalization trough lexical similarities constrained in them. This allows for automatically generating feature spaces of generalized syntactic/semantic dependency substructures.

To test our models, we carried out experiments on QC and SRL. These show that by exploiting the similarity between two sets of words carried out according to their dependency structure leads to an unprecedented result for QC, i.e. 94.8% of accuracy. In contrast, when no structure is used the accuracy does not significantly improves. We have also provided a fast algorithm for the computation of SPTK and empirically shown that it can easily scale.

It should be noted that our models are not absolutely restricted to QC and SRL. Indeed, since most of the NLP applications are based on syntactic and lexical representations, SPTK will have a major impact in most of them, e.g.:

- Question Answering, the high results for QC will positively impact on the overall task.
- SRL, SPTK alone reaches the state-of-the-art (SOA) (only 0.7% less) in FrameNet role classification. This is very valuable as previous work showed that tree kernels (TK) alone perform lower than models based on manually engineered features for SRL tasks, e.g., (Moschitti, 2004; Giuglea and Moschitti, 2004; Giuglea and Moschitti, 2006; Moschitti, 2006b; Che et al., 2006; Moschitti et al., 2008). Thus for the first time in an SRL task, a general tree kernel reaches the same accuracy of heavy manual feature design. This also suggests an improvement when used in combinations with manual feature vectors.
- Relation Extraction and Pronominal Coreference, whose state-of-the-art for some tasks is achieved with the simple STK-CT (see (Zhang et al., 2006) and (Yang et al., 2006; Versley et al., 2008), respectively).
- In word sense disambiguation tasks, SPTK can generalize context according to syntactic and semantic constraints (selectional restrictions) making very effective distributional semantic approaches.
- In Opinion Mining SPTK will allow to match sentiment words within their corresponding syntactic counterparts and improve the state-of-the-art (Johansson and Moschitti, 2010b; Johansson and Moschitti, 2010a).
- Experiments on Recognizing Textual Entailment (RTE) tasks, the use of SSTK (instead of STK-CT) improved the state-of-the-art (Mehdad et al., 2010). SPTK may provide further enhancement and innovative and effective dependency models.

The above points also suggest many promising future research directions, which we would like to explore.

Acknowledgements

This work has been partially supported by the EC project FP247758: Trustworthy Eternal Systems via Evolving Software, Data and Knowledge (EternalS).

[6]Using one of the 8 processors of an Intel(R) Xeon(R) CPU E5430 @ 2.66GHz machine, 32Gb Ram.

References

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of CoNLL-2005*, pages 1–8, Ann Arbor, Michigan. Association for Computational Linguistics.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In *Proceedings of ECIR 2007, Rome, Italy*.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *In Proceedings of CIKM '07*.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of ICDM 06, Hong Kong, 2006*.

Ulrik Brandes. 2001. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25:163–177.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT and EMNLP*, pages 724–731, Vancouver, British Columbia, Canada, October.

Horst Bunke and Kim Shearer. 1998. A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, 19(3-4):255–259, March.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

O. Chapelle, B. Schlkopf, and A. Zien. 2006. *Semi-Supervised Learning*. Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA, 09.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL'00*.

Wanxiang Che, Min Zhang, Ting Liu, and Sheng Li. 2006. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 73–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL'02*.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jim Cowie, Joe Guthrie, and Louise Guthrie. 1992. Lexical disambiguation using simulated annealing. In *in COLING*, pages 359–365.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. 2001. Latent semantic kernels. In Carla Brodley and Andrea Danyluk, editors, *Proceedings of ICML-01, 18th International Conference on Machine Learning*, pages 66–73, Williams College, US. Morgan Kaufmann Publishers, San Francisco, US.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*, pages 423–429, Barcelona, Spain, July.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Hal Daumé III and Daniel Marcu. 2004. Np bracketing by maximum entropy tagging and SVM reranking. In *Proceedings of EMNLP'04*.

Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. 2007. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 209–216, New York, NY, USA. ACM.

Linton C. Freeman. 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41.

Hagen Fürstenau and Mirella Lapata. 2009. Graph alignment for semi-supervised semantic role labeling. In *In Proceedings of EMNLP '09*, pages 11–20, Morristown, NJ, USA.

Ana-Maria Giuglea and Alessandro Moschitti. 2004. Knowledge Discovering using FrameNet, VerbNet and PropBank. In *In Proceedings of the Workshop on Ontology and Knowledge Discovering at ECML 2004, Pisa, Italy*.

A.-M. Giuglea and A. Moschitti. 2006. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of ACL*, Sydney, Australia.

Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of ACL'05*, pages 403–410.

G. Golub and W. Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 2(2):pp. 205–224.

Zellig Harris. 1964. Distributional structure. In Jerrold J. Katz and Jerry A. Fodor, editors, *The Philosophy of Linguistics*. Oxford University Press.

J. J. Jiang and D. W. Conrath. 1997. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*.

T. Joachims. 2000. Estimating the generalization performance of a SVM efficiently. In *Proceedings of ICML'00*.

Richard Johansson and Alessandro Moschitti. 2010a. Reranking models in fine-grained opinion analysis. In *Proceedings of the 23rd International Conference of Computational Linguistics (Coling 2010)*, pages 519–527, Beijing, China.

Richard Johansson and Alessandro Moschitti. 2010b. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 67–76, Uppsala, Sweden.

Richard Johansson and Pierre Nugues. 2008a. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 183–187, Manchester, United Kingdom.

Richard Johansson and Pierre Nugues. 2008b. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING*, Manchester, UK, August 18-22.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

Claudia Leacock and Martin Chodorow, 1998. *Combining Local Context and WordNet Similarity for Word Sense Identification*, chapter 11, pages 265–283. The MIT Press.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of ACL'02*.

Yashar Mehdad, Alessandro Moschitti, and Fabio Massimo Zanzotto. 2010. Syntactic/semantic structures for textual entailment recognition. In *HLT-NAACL*, pages 1020–1028.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2005. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the American Association for Artificial Intelligence (AAAI 2006)*, Boston, July.

Rada Mihalcea. 2005. unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *In HLT/EMNLP 2005*, pages 411–418.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

A. Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of ACL*, Barcelona, Spain.

Alessandro Moschitti. 2006a. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of ECML'06*, pages 318–329.

Alessandro Moschitti. 2006b. Making tree kernels practical for natural language learning. In *Proccedings of EACL'06*.

Roberto Navigli and Mirella Lapata. 2010. An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692.

Sebastian Pado and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2).

Sebastian Padó, 2006. *User's guide to* `sigf`*: Significance testing by approximate randomisation*.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004a. WordNet::Similarity - Measuring the Relatedness of Concept. In *Proc. of 5th NAACL*, Boston, MA.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004b. Wordnet::similarity - measuring the relatedness of concepts. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Demonstration Papers*, pages 38–41, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

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

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.

Hinrich Schtze. 1998. Automatic word sense discrimination. *Journal of Computational Linguistics*, 24:97–123.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Libin Shen, Anoop Sarkar, and Aravind k. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Empirical Methods for Natural Language Processing (EMNLP)*, pages 89–96, Sapporo, Japan.

Georges Siolas and Florence d'Alch Buc. 2000. Support vector machines based on a semantic kernel for

text categorization. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 5*, page 5205. IEEE Computer Society.

Ivan Titov and James Henderson. 2006. Porting statistical parsers with data-defined kernels. In *Proceedings of CoNLL-X*.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008. Coreference systems based on kernels methods. In *The 22nd International Conference on Computational Linguistics (Coling'08)*, Manchester, England.

Zhibiao Wu and Martha Palmer. 1994. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138, New Mexico State University, Las Cruces, New Mexico.

Xiaofeng Yang, Jian Su, and Chewlim Tan. 2006. Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. COLING-ACL 06*.

Alexander S. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING*, pages 947–953.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of EMNLP-ACL*, pages 181–201.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM Press.

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring Syntactic Features for Relation Extraction using a Convolution tree kernel. In *Proceedings of NAACL*.

Peixiang Zhao, Jiawei Han, and Yizhou Sun. 2009. P-Rank: a comprehensive structural similarity measure over information networks. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 553–562, New York, NY, USA. ACM.

Probabilistic models of similarity in syntactic context

Diarmuid Ó Séaghdha
Computer Laboratory
University of Cambridge
United Kingdom
do242@cl.cam.ac.uk

Anna Korhonen
Computer Laboratory
University of Cambridge
United Kingdom
Anna.Korhonen@cl.cam.ac.uk

Abstract

This paper investigates novel methods for incorporating syntactic information in probabilistic latent variable models of lexical choice and contextual similarity. The resulting models capture the effects of context on the interpretation of a word and in particular its effect on the appropriateness of replacing that word with a potentially related one. Evaluating our techniques on two datasets, we report performance above the prior state of the art for estimating sentence similarity and ranking lexical substitutes.

1 Introduction

Distributional models of lexical semantics, which assume that aspects of a word's meaning can be related to the contexts in which that word is typically used, have a long history in Natural Language Processing (Spärck Jones, 1964; Harper, 1965). Such models still constitute one of the most popular approaches to lexical semantics, with many proven applications. Much work in distributional semantics treats words as non-contextualised units; the models that are constructed can answer questions such as "how similar are the words *body* and *corpse*?" but do not capture the way the syntactic context in which a word appears can affect its interpretation. Recent developments (Mitchell and Lapata, 2008; Erk and Padó, 2008; Thater et al., 2010; Grefenstette et al., 2011) have aimed to address compositionality of meaning in terms of distributional semantics, leading to new kinds of questions such as "how similar are the usages of the words *body* and *corpse* in the phrase *the body/corpse deliberated the motion*...?" and "how similar are the phrases *the body deliberated the motion* and *the corpse rotted*?". In this paper we focus on answering questions of the former type and investigate models that describe the effect of syntactic context on the meaning of a single word.

The work described in this paper uses probabilistic latent variable models to describe patterns of syntactic interaction, building on the selectional preference models of Ó Séaghdha (2010) and Ritter et al. (2010) and the lexical substitution models of Dinu and Lapata (2010). We propose novel methods for incorporating information about syntactic context in models of lexical choice, yielding a probabilistic analogue to dependency-based models of contextual similarity. Our models attain state-of-the-art performance on two evaluation datasets: a set of sentence similarity judgements collected by Mitchell and Lapata (2008) and the dataset of the English Lexical Substitution Task (McCarthy and Navigli, 2009). In view of the well-established effectiveness of dependency-based distributional semantics and of probabilistic frameworks for semantic inference, we expect that our approach will prove to be of value in a wide range of application settings.

2 Related work

The literature on distributional semantics is vast; in this section we focus on outlining the research that is most directly related to capturing effects of context and compositionality.[1] Mitchell and Lapata (2008)

[1] The interested reader is referred to Padó and Lapata (2007) and Turney and Pantel (2010) for a general overview.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1047–1057,
Edinburgh, Scotland, UK, July 27–31, 2011.

follow Kintsch (2001) in observing that most distributional approaches to meaning at the phrase or sentence level assume that the contribution of syntactic structure can be ignored and the meaning of a phrase is simply the commutative sum of the meanings of its constituent words. As Mitchell and Lapata argue, this assumption clearly leads to an impoverished model of semantics. Mitchell and Lapata investigate a number of simple methods for combining distributional word vectors, concluding that pointwise multiplication best corresponds to the effects of syntactic interaction.

Erk and Padó (2008) introduce the concept of a *structured vector space* in which each word is associated with a set of selectional preference vectors corresponding to different syntactic dependencies. Thater et al. (2010) develop this geometric approach further using a space of *second-order* distributional vectors that represent the words typically co-occurring with the contexts in which a word typically appears. The primary concern of these authors is to model the effect of context on word meaning; the work we present in this paper uses similar intuitions in a probabilistic modelling framework.

A parallel strand of research seeks to represent the meaning of larger compositional structures using matrix and tensor algebra (Smolensky, 1990; Rudolph and Giesbrecht, 2010; Baroni and Zamparelli, 2010; Grefenstette et al., 2011). This nascent approach holds the promise of providing a much richer notion of context than is currently exploited in semantic applications.

Probabilistic latent variable frameworks for generalising about contextual behaviour (in the form of verb-noun selectional preferences) were proposed by Pereira et al. (1993) and Rooth et al. (1999). Latent variable models are also conceptually similar to non-probabilistic dimensionality reduction techniques such as Latent Semantic Analysis (Landauer and Dumais, 1997). More recently, Ó Séaghdha (2010) and Ritter et al. (2010) reformulated Rooth et al.'s approach in a Bayesian framework using models related to Latent Dirichlet Allocation (Blei et al., 2003), demonstrating that this "topic modelling" architecture is a very good fit for capturing selectional preferences. Reisinger and Mooney (2010) investigate nonparametric Bayesian models for teasing apart the context distributions of polysemous words. As described in Section 3 below, Dinu and Lapata (2010) propose an LDA-based model for lexical substitution; the techniques presented in this paper can be viewed as a generalisation of theirs. Topic models have also been applied to other classes of semantic task, for example word sense disambiguation (Li et al., 2010), word sense induction (Brody and Lapata, 2009) and modelling human judgements of semantic association (Griffiths et al., 2007).

3 Models

3.1 Latent variable context models

In this paper we consider generative models of lexical choice that assign a probability to a particular word appearing in a given linguistic context. In particular, we follow recent work (Dinu and Lapata, 2010; Ó Séaghdha, 2010; Ritter et al., 2010) in assuming a latent variable model that associates contexts with distributions over a shared set of variables and associates each variable with a distribution over the vocabulary of word types:

$$P(w|c) = \sum_{z \in Z} P(w|z)P(z|c) \tag{1}$$

The set of latent variables Z is typically much smaller than the vocabulary size; this induces a (soft) clustering of the vocabulary. Latent Dirichlet Allocation (Blei et al., 2003) is a powerful method for learning such models from a text corpus in an unsupervised way; LDA was originally applied to document modelling, but it has recently been shown to be very effective at inducing models for a variety of semantic tasks (see Section 2).

Given the latent variable framework in (1) we can develop a generative model of paraphrasing a word o with another word n in a particular context c:

$$P_{C \to T}(n|o,c) = \sum_{z} P(n|z)P(z|o,c) \tag{2}$$

$$P(z|o,c) = \frac{P(o|z)P(z|c)}{\sum_{z'} P(o|z')P(z'|c)} \tag{3}$$

In words, the probability $P(n|o,c)$ is the probability that n would be generated given the latent variable distribution associated with seeing o in context c; this latter distribution $P(z|o,c)$ can be derived using Bayes' rule and the assumption $P(o|z,c) = P(o|z)$.

Given a set of contexts C in which an instance o appears (e.g., it may be both the subject of a verb and modified by an adjective), (2) and (3) become:

$$P_{C\to T}(n|o,C) = \sum_z P(n|z)P(z|o,C) \quad (4)$$

$$P(z|o,C) = \frac{P(o|z)P(z|C)}{\sum_{z'} P(o|z')P(z'|C)} \quad (5)$$

$$P(z|C) = \frac{\prod_{c\in C} P(z|c)}{\sum_{z'} \prod_{c\in C} P(z'|c)} \quad (6)$$

Equation (6) can be viewed as defining a "product of experts" model (Hinton, 2002). Dinu and Lapata (2010) also use a similar formulation to (5), except that $P(z|o,C)$ is factorised over $P(z|o,C)$ rather than just $P(z|C)$:

$$P_{DL10}(z|o,C) = \prod_{c\in C} \frac{P(o|z)P(z|c)}{\sum_{z'} P(o|z')P(z'|c)} \quad (7)$$

In Section 5 below, we find that using (5) rather than (7) gives better results.

The model described above (henceforth $C \to T$) models the dependence of a target word on its context. An alternative perspective is to model the dependence of a set of contexts on a target word, i.e., we induce a model

$$P(c|w) = \sum_z P(c|z)P(z|w) \quad (8)$$

Making certain assumptions, a formula for $P(n|o,c)$ can be derived from (8):

$$P_{T\to C}(n|o,c) = \frac{P(c|o,n)P(n|o)}{P(c|o)} \quad (9)$$

$$P(c|o,n) = \sum_z P(c|z)P(z|o,n)$$

$$P(z|o,n) = \frac{P(z|o)P(z|n)}{\sum_{z'} P(z'|o)P(z'|n)} \quad (10)$$

$$P(c|o) = \sum_z P(c|z)P(z|o) \quad (11)$$

$$P(n|o) = 1/V \quad (12)$$

The assumption of a uniform prior $P(n|o)$ on the choice of a paraphrase n for o is clearly not appropriate from a language modelling perspective (one could imagine an alternative $P(n)$ based on corpus frequency), but in the context of measuring semantic similarity it serves well. The $T \to C$ model for a set of contexts C is:

$$P_{T\to C}(n|o,C) = \frac{P(C|o,n)P(n|o)}{P(C|o)} \quad (13)$$

$$P(C|o,n) = \sum_z P(z|o,n) \prod_{c\in C} P(c|z) \quad (14)$$

$$P(C|o) = \sum_z P(z|o) \prod_{c\in C} P(c|z) \quad (15)$$

$$P(z|o,C) = \frac{P(z|o)P(C|o)}{\sum_{z'} P(z'|o)P(C|o)} \quad (16)$$

With appropriate priors chosen for the distributions over words and latent variables, $P(n|o,C)$ is a fully generative model of lexical substitution. A non-generative alternative is one that estimates the similarity of the latent variable distributions associated with seeing n and o in context C. The principle that similarity between topic distributions corresponds to semantic similarity is well-known in document modelling and was proposed in the context of lexical substitution by Dinu and Lapata (2010). In terms of the equations presented above, we could compare the distributions $P(\mathbf{z}|o,C)$ with $P(\mathbf{z}|n,C)$ using equations (5) or (16). However, Thater et al. (2010) and Dinu and Lapata (2010) both observe that contextualising both o and n can degrade performance; in view of this we actually compare $P(\mathbf{z}|o,C)$ with $P(\mathbf{z}|n)$ and make the further simplifying assumption that $P(z|n) \propto P(n|z)$. The similarity measure we adopt is the Bhattacharyya coefficient, which is a natural measure of similarity between probability distributions and is closely related to the Hellinger distance used in previous work on topic modelling (Blei and Lafferty, 2007):

$$sim_{bhatt}(P_x(\mathbf{z}), P_y(\mathbf{z})) = \sum_z \sqrt{P_x(z)P_y(z)} \quad (17)$$

This measure takes values between 0 and 1.

In this paper we train LDA models of $P(w|c)$ and $P(c|w)$. In the former case, the analogy to document modelling is that each context type plays the role of a "document" consisting of all the words observed in that context in a corpus; for $P(c|w)$ the roles are reversed. The models are trained by Gibbs sampling using the efficient procedure of Yao et al. (2009). The empirical estimates for distributions over words and latent variables are derived from the assignment

of topics over the training corpus in a single sampling state. For example, to model $P(w|c)$ we calculate:

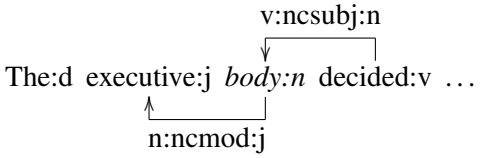
$$P(w|z) = \frac{f_{zw} + \beta}{f_{z\cdot} + N\beta} \quad (18)$$

$$P(z|c) = \frac{f_{zc} + \alpha_z}{f_{\cdot c} + \sum_{z'} \alpha_{z'}} \quad (19)$$

where f_{zw} is the number of words of type w assigned topic z, f_{zc} is the number of times z is associated with context c, $f_{z\cdot}$ and $f_{\cdot c}$ are the marginal topic and context counts respectively, N is the number of word types and α and β parameterise the Dirichlet prior distributions over $P(z|c)$ and $P(w|z)$. Following the recommendations of Wallach et al. (2009) we use asymmetric α and symmetric β; rather than using fixed values for these hyperparameters we estimate them from data in the course of LDA training using an EM-like method.[2] We use standard settings for the number of training iterations (1000), the length of the burnin period before hyperparameter estimation begins (200 iterations) and the frequency of hyperparameter estimation (50 iterations).

3.2 Context types

We have not yet defined what the contexts c look like. In vector space models of semantics it is common to distinguish between window-based and dependency-based models (Padó and Lapata, 2007); one can make the same distinction for probabilistic context models. A broad generalisation is that window-based models capture semantic association (e.g. *referee* is associated with *football*), while dependency models capture a finer-grained notion of similarity (*referee* is similar to *umpire* but not to *football*). Dinu and Lapata (2010) propose a window-based model of lexical substitution; the set of contexts in which a word appears is the set of surrounding words within a prespecified "window size". In this paper we also investigate dependency-based context sets derived from syntactic structure. Given a sentence such as



the set C of dependency contexts for the noun *body* is $\{$*executive:j:ncmod*$^{-1}$*:n*, *decide:v:ncsubj:n*$\}$, where $ncmod^{-1}$ denotes that *body* stands in an inverse non-clausal modifier relation to *executive* (we assume that nouns are the heads of their adjectival modifiers).

4 Experiment 1: Similarity in context

4.1 Data

Mitchell and Lapata (2008) collected human judgements of semantic similarity for pairs of short sentences, where the sentences in a pair share the same subject but different verbs. For example, *the sales slumped* and *the sales declined* should be judged as very similar while *the shoulders slumped* and *the shoulders declined* should be judged as less similar. The resulting dataset (henceforth ML08) consists of 120 such pairs using 15 verbs, balanced across high and low expected similarity. 60 subjects rated the data using a scale of 1–7; Mitchell and Lapata calculate average interannotator correlation to be 0.40 (using Spearman's ρ). Both Mitchell and Lapata and Erk and Padó (2008) split the data into a development portion and a test portion, the development portion consisting of the judgements of six annotators; in order to compare our results with previous research we use the same data split. To evaluate performance, the predictions made by a model are compared to the judgements of each annotator in turn (using ρ) and the resulting per-annotator ρ values are averaged.

4.2 Models

All models were trained on the written section of the British National Corpus (around 90 million words), parsed with RASP (Briscoe et al., 2006). The BNC was also used by Mitchell and Lapata (2008) and Erk and Padó (2008); as the ML08 dataset was compiled using words appearing more than 50 times in the BNC, there are no coverage problems caused by data sparsity. We trained LDA models for the grammatical relations *v:ncsubj:n* and *n:ncsubj*$^{-1}$*:v*

[2] We use the estimation methods provided by the MALLET toolkit, available from `http://mallet.cs.umass.edu/`.

	Model	PARA	SIM
No optimisation	C → T	0.24	0.34
	T → C	0.36	**0.39**
	T ↔ C	0.33	0.39
Optimised on dev	C → T	0.24	0.35
	T → C	**0.41**	**0.41**
	T ↔ C	0.37	**0.41**
Erk and Padó (2008)	Mult	0.24	
	SVS	0.27	

Table 1: Performance (average ρ) on the ML08 test set

and used these to create predictors of type $C \rightarrow T$ and $T \rightarrow C$, respectively. For each predictor, we trained five runs with 100 topics for 1000 iterations and averaged the predictions produced from their final states. We investigate both the generative paraphrasing model (PARA) and the method of comparing topic distributions (SIM). For both PARA and SIM we present results using each predictor type on its own as well as a combination of both types ($T \leftrightarrow C$); for PARA the contributions of the types are multiplied and for SIM they are averaged.[3] One potential complication is that the PARA model is trained to predict $P(n|c, o)$, which might not be comparable across different combinations of subject c and verb o. Using $P(n|c, o)$ as a proxy for the desired joint distribution $P(n, c, o)$ is tantamount to assuming a uniform distribution $P(c, o)$, which can be defended on the basis that the choice of subject noun and reference verb is not directly relevant to the task. As shown by the results below, this assumption seems to work reasonably well in practice.

As well as reporting correlations for straightforward averages of each set of five runs, we also investigate whether the development data can be used to select an optimal subset of runs. This is done by simply evaluating every possible subset of 1–5 runs on the development data and picking the best-scoring subset.

4.3 Results

Table 1 presents the results of the PARA and SIM predictors on the ML08 dataset. The best results previously reported for this dataset were given by Erk and Padó (2008), who measured average ρ values of 0.24 for a vector multiplication method and 0.27 for their *structured vector space* (SVS) syntactic disambiguation method. Even without using the development set to select models, performance is well above the previous state of the art for all predictors except $\text{PARA}_{C \rightarrow T}$. Model selection on the development data brings average ρ up to 0.41, which is comparable to the human "ceiling" of 0.40 measured by Mitchell and Lapata. In all cases the $T \rightarrow C$ predictors outperform $C \rightarrow T$: models that associate target words with distributions over context clusters are superior to those that associate contexts with distributions over target words.

Figure 1 plots the beneficial effect of averaging over multiple runs; as the number of runs n is increased, the average performance over all combinations of n predictors chosen from the set of five $T \rightarrow C$ and five $C \rightarrow T$ runs is observed to increase monotonically. Figure 1 also shows that the model selection procedure is very effective at selecting the optimal combination of models; development set performance is a reliable indicator of test set performance.

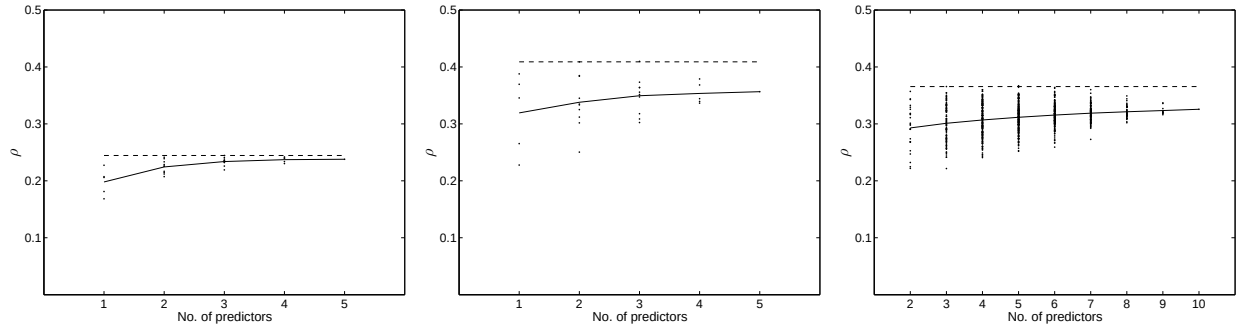
5 Experiment 2: Lexical substitution

5.1 Data

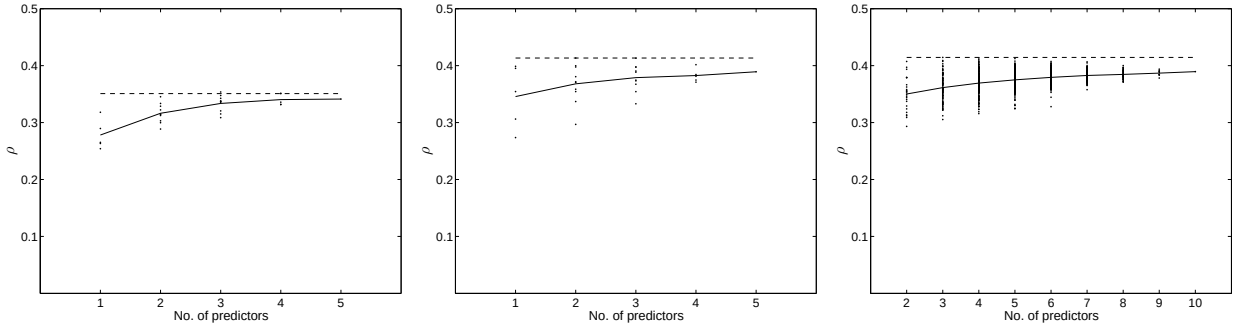
The English Lexical Substitution task, run as part of the SemEval-1 competition, required participants to propose good substitutes for a set of target words in various sentential contexts (McCarthy and Navigli, 2009). Table 2 shows two example sentences and the substitutes appearing in the gold standard, ranked by the number of human annotators who proposed each substitute. The dataset contains a total of 2,010 annotated sentences with 205 distinct target words across four parts of speech (noun, verb, adjective, adverb). In line with previous work on contextual disambiguation, we focus here on the subtask of ranking attested substitutes rather than proposing them from an unrestricted vocabulary. To this end, a candidate set is constructed for each target word from all the substitutes proposed for that word in all sentences in the dataset.

The data contains a number of multiword paraphrases such as *rush at*; as our models (like most

[3] This configuration seems the most intuitive; averaging PARA predictors and multiplying SIM also give good results.



(a) PARA: Target → Context (b) PARA: Context → Target (c) PARA: Target ↔ Context



(d) SIM: Target → Context (e) SIM: Context → Target (f) SIM: Target ↔ Context

Figure 1: Performance on the ML08 test set with different predictor types and different numbers of LDA runs per predictor type; the solid line tracks the average performance, the dashed line shows the performance of the predictor combination that scores best on the development set.

Realizing immediately that strangers have come, the animals *charge* them and the horses began to fight.	attack (5), rush at (1)
Commission is the amount *charged* to execute a trade.	levy (2), impose (1), take (1), demand (1)

Table 2: Examples for the verb *charge* from the English Lexical Substitution Task

current models of distributional semantics) do not represent multiword expressions, we remove such paraphrases and discard the 17 sentences which have only multiword substitutes in the gold standard.[4] There are also 7 sentences for which the gold standard contains no substitutes. This leaves a total of 1986 sentences. These sentences were lemmatised and parsed with RASP.

Previous authors have partitioned the dataset in various ways. Erk and Padó (2008) use only a subset of the data where the target is a noun headed by a verb or a verb heading a noun. Thater et al. (2010) discard sentences which their parser cannot parse and paraphrases absent from their training corpus and then optimise the parameters of their model through four-fold cross-validation. Here we aim for complete coverage on the dataset and do not perform any parameter tuning. We use two measures to evaluate performance: Generalised Averaged Precision (Kishida, 2005) and Kendall's τ_b rank correlation coefficient, which were used for this task by Thater et al. (2010) and Dinu and Lapata (2010), respectively. Generalised Averaged Precision (GAP) is a precision-like measure for evaluating ranked predictions against a gold standard. τ_b is a variant of Kendall's τ that is appropriate for data containing tied ranks. We do not use the "precision out of ten"

[4]Thater et al. (2010) and Dinu and Lapata (2010) similarly remove multiword paraphrases (Georgiana Dinu, p.c.).

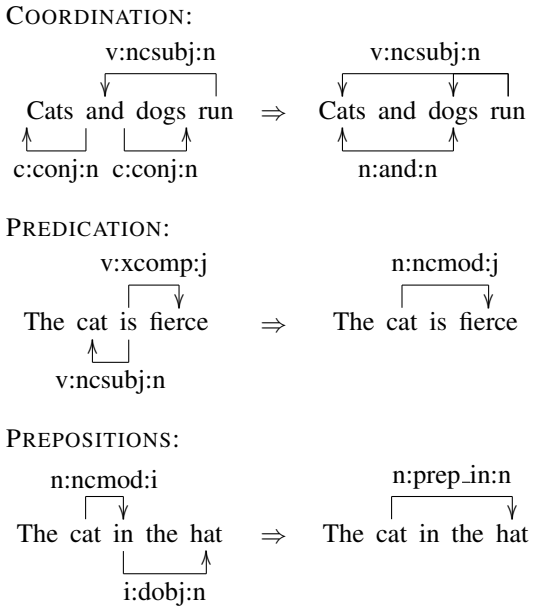


Table 3: Dependency graph preprocessing

measure that was used in the original Lexical Substitution Task; this measure assigns credit for the proportion of the first 10 proposed paraphrases that are present in the gold standard and in the context of ranking attested substitutes it is unclear how to obtain non-trivial results for target words with 10 or fewer possible substitutes. We calculate statistical significance of performance differences using stratified shuffling (Yeh, 2000).[5]

5.2 Models

We apply the models developed in Section 3.1 to the Lexical Substitution Task dataset using dependency- and window-based context information. Here we only use the SIM predictor type. PARA did not give satisfactory results; in particular, it tended to rank common words highly in most contexts.[6]

As before we compiled training data by extracting target-context cooccurrences from a text corpus. In addition to the parsed BNC described above we used a corpus of Wikipedia text consisting of over 45 million sentences (almost 1 billion words) parsed using the fast Combinatory Categorial Grammar (CCG) parser described by Clark et al. (2009). The dependency representation produced by this parser is interoperable with the RASP dependency format. In order to focus our models on semantically discriminative information and make inference more tractable we ignored all parts of speech other than nouns, verbs, adjectives, prepositions and adverbs. Stopwords and words of fewer than three characters were removed. We also removed the very frequent but semantically weak lemmas *be* and *have*.

We compare two classes of context models: models learned from window-based contexts and models learned from syntactic dependency contexts. For the syntactic models we extracted all dependencies and inverse dependencies between lemmas of the aforementioned POS types; in order to maximise the extraction yield, the dependency graph for each sentence was preprocessed using the transformations shown in Table 3. For the window-based context model we follow Dinu and Lapata (2010) in treating each word within five words of a target as a member of its context set.

It proved necessary to subsample the corpora in order to make LDA training tractable, especially for the window-based model where the training set of context-target counts is extremely dense (each instance of a word in the corpus contributes up to 10 context instances). For the window-based data, we divided each context-target count by a factor of 5 and a factor of 70 for the BNC and Wikipedia corpora respectively, rounding fractional counts to the closest integer. The choice of 70 for scaling Wikipedia counts is adopted from Dinu and Lapata (2010), who used the same factor for the comparably sized English Gigaword corpus. As the dependency data is an order of magnitude smaller we downsampled the Wikipedia counts by 5 and left the BNC counts untouched. Finally, we created a larger corpus by combining the counts from the BNC and Wikipedia datasets. Type and token counts for the BNC and combined corpora are given in Table 4.

We trained three LDA predictors for each corpus: a window-based predictor (W5), a Context $\rightarrow$ Target predictor ($C \rightarrow T$) and a Target $\rightarrow$ Context predictor ($T \rightarrow C$). For W5 the sets of types and contexts should be symmetrical (in practice there is some discrepancy due to preprocessing artefacts). For $C \rightarrow T$, individual models were trained for each of the four target parts of speech; in each case the set

[5] We use the software package available at `http://www.nlpado.de/~sebastian/sigf.html`.

[6] Favouring more general words may indeed make sense in some paraphrasing tasks (Nulty and Costello, 2010).

	BNC			BNC+Wikipedia		
	Tokens	Types	Contexts	Tokens	Types	Contexts
Nouns	18723082	122999	316237	54145216	106448	514257
Verbs	7893462	18494	57528	20082658	16673	82580
Adjectives	4385788	73684	37163	11536424	88488	57531
Adverbs	1976837	7124	14867	3017936	4056	18510
Window5	28329238	88265	102792	42828094	139640	143443

Table 4: Type and token counts for the BNC and downsampled BNC+Wikipedia corpora

	BNC			BNC + Wikipedia		
	GAP	τ_b	Coverage	GAP	τ_b	Coverage
W5	44.5	0.17	100.0	44.8	0.17	100.0
$C \rightarrow T$	43.2	0.16	86.4	48.7	0.21	86.5
$T \rightarrow C$	47.2	0.21	86.4	49.3	0.22	86.5
$T \leftrightarrow C$	45.7	0.20	86.4	49.1	0.23	86.5
W5 + $C \rightarrow T$	46.0	0.18	100.0	48.7	0.21	100.0
W5 + $T \rightarrow C$	**48.6**	**0.21**	100.0	49.3	0.22	100.0
W5 + $T \leftrightarrow C$	48.1	0.20	100.0	**49.5**	**0.23**	100.0

Table 5: Results on the English Lexical Substitution Task dataset; boldface denotes best performance at full coverage for each corpus

of types is the vocabulary for that part of speech and the set of contexts is the set of dependencies taking those types as dependents. For $T \rightarrow C$ we again train four models; the sets of types and contexts are reversed. For the both corpora we trained models with $Z = \{600, 800, 1000, 1200\}$ topics; for each setting of Z we ran five estimation runs. Each individual prediction of similarity between $P(z|C, o)$ and $P(z|n)$ is made by averaging over the predictions of all runs and over all settings of Z. Choosing a single setting of Z does not degrade performance significantly; however, averaging over settings is a convenient way to avoid having to pick a specific value.

We also investigate combinations of predictor types, once again produced by averaging: we combine $C \rightarrow T$ with $C \leftrightarrow T$ ($T \leftrightarrow C$) and combine each of these three models with W5.

5.3 Results

Table 5 presents the results attained by our models on the Lexical Substitution Task data. The dependency-based models have imperfect coverage (86% of the data); they can make no prediction when no syntactic context is provided for a target, perhaps as a result of parsing error. The window-based models have perfect coverage, but score noticeably lower. By combining dependency- and window-based models we can reach high performance with perfect coverage. All combinations outperform the corresponding W5 results to a statistically significant degree ($p < 0.01$). Performance at full coverage is already very good (GAP= 48.6, $\tau_b = 0.21$) on the BNC corpus, but the best results are attained by W5 + $T \leftrightarrow C$ trained on the combined corpus (GAP= 49.5, $\tau_b = 0.23$). The results for the W5 model trained on BNC data is comparable to that trained on the combined corpus; however the syntactic models show a clear benefit from the less sparse dependency data in the combined training corpus.

As remarked in Section 3.1, Dinu and Lapata (2010) use a slightly different formulation of $P(z|C, o)$. Using the window-based context model our formulation (5) outperforms (7) for both training corpora; the Dinu and Lapata (2010) version scores GAP $= 41.5, \tau_b = 0.15$ for the BNC corpus and GAP $= 42.0, \tau_b = 0.15$ for the combined corpus. The advantage of our formulation is statistically significant for all evaluation measures.

	Nouns		Verbs		Adjectives		Adverbs		Overall	
	GAP	τ_b	GAP	τ_b	GAP	τ_b	GAP	τ_b	GAP	τ_b
W5	46.0	0.16	38.9	0.14	44.0	0.18	54.0	0.22	44.8	0.17
W5 + $T \leftrightarrow C$	**50.7**	**0.22**	45.1	**0.20**	**48.8**	**0.24**	**55.9**	0.24	**49.5**	**0.23**
Thater et al. (2010) (Model 1)	46.4	–	**45.9**	–	39.4	–	48.2	–	44.6	–
Thater et al. (2010) (Model 2)	42.5	–	–	–	43.2	–	51.4	–	–	–
Dinu and Lapata (2010) (LDA)	–	0.16	–	0.14	–	0.17	–	0.21	–	0.16
Dinu and Lapata (2010) (NMF)	–	0.15	–	0.14	–	0.16	–	**0.26**	–	0.16

Table 6: Performance by part of speech

Table 6 gives a breakdown of performance by target part of speech for the BNC+Wikipedia-trained W5 and W5 + $T \leftrightarrow C$ models, as well as figures provided by previous researchers.[7] W5 + $T \leftrightarrow C$ outperforms W5 on all parts of speech using both evaluation metrics. As remarked above, previous researchers have used the corpus in slightly different ways; we believe that the results of Dinu and Lapata (2010) are fully comparable, while those of Thater et al. (2010) were attained on a slightly smaller dataset with parameters set through cross-validation. The results for W5 + $T \leftrightarrow C$ outperform all of Dinu and Lapata's per-POS and overall results except for a slightly superior score on adverbs attained by their NMF model ($\tau_b = 0.26$ compared to 0.24). Turning to Thater et al., we report higher scores for every POS with the exception of the verbs where their Model 1 achieves 45.9 GAP compared to 45.1; the overall average for W5 + $T \leftrightarrow C$ is substantially higher at 49.5 compared to 44.6. On balance, we suggest that our models do have an advantage over the current state of the art for lexical substitution.

6 Conclusion

In this paper we have proposed novel methods for modelling the effect of context on lexical meaning, demonstrating that information about syntactic context and textual proximity can fruitfully be integrated to produce state-of-the-art models of lexical choice. We have demonstrated the effectiveness of our techniques on two datasets but they are potentially applicable to a range of applications where semantic disambiguation is required. In future work, we intend to adapt our approach for word sense disambiguation as well as related domain-specific tasks such as gene name normalisation (Morgan et al., 2008). A further, more speculative direction for future research is to investigate more richly structured models of context, for example capturing correlations between words in a text within a framework similar to the Correlated Topic Model of Blei and Lafferty (2007) or more explicitly modelling polysemy effects as in Reisinger and Mooney (2010).

[7] The overall average GAP for Thater et al. (2010) does not appear in their paper but can be calculated from the score and number of instances listed for each POS.

Acknowledgements

We are grateful to the EMNLP reviewers for their helpful comments. This research was supported by EPSRC grant EP/G051070/1.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, Cambridge, MA.

David M. Blei and John D. Lafferty. 2007. A correlated topic model of science. *The Annals of Applied Statistics*, 1(1):17–35.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the ACL-06 Interactive Presentation Sessions*, Sydney, Australia.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of EACL-09*, Athens, Greece.

Stephen Clark, Ann Copestake, James R. Curran, Yue Zhang, Aurelie Herbelot, James Haggerty, Byung-Gyu Ahn, Curt Van Wyk, Jessika Roesner, Jonathan Kummerfeld, and Tim Dawborn. 2009. Large-scale syntactic processing: Parsing the web. Technical report, Final Report of the 2009 JHU CLSP Workshop.

Georgiana Dinu and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, Cambridge,MA.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, Honolulu, HI.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. 2011. Concrete sentence spaces for compositional distributional models of meaning. In *Proceedings of the 9th International Conference on Computational Semantics (IWCS-11)*, Oxford, UK.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.

Kenneth E. Harper. 1965. Measurement of similarity between nouns. In *Proceedings of the 1965 International Conference on Computational Linguistics (COLING-65)*, New York, NY.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Walter Kintsch. 2001. Predication. *Cognitive Science*, 25(2):173–202.

Kazuaki Kishida. 2005. Property of average precision and its generalisation: An examination of evaluation indicator for information retrieval experiments. Technical Report NII-2005-014E, National Institute of Informatics, Tokyo, Japan.

Thomas K Landauer and Susan T Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

Diana McCarthy and Roberto Navigli. 2009. The English lexical substitution task. *Language Resources and Evaluation*, 43(2):139–159.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, Columbus, OH.

Alexander A. Morgan, Zhiyong Lu, Xinglong Wang, Aaron M Cohen, Juliane Fluck, Patrick Ruch, Anna Divoli, Katrin Fundel, Robert Leaman, Jörg Hakenberg, Chengjie Sun, Heng hui Liu, Rafael Torres, Michael Krauthammer, William W Lau, Hongfang Liu, Chun-Nan Hsu, Martijn Schuemie, K. Bretonnel Cohen, and Lynette Hirschman. 2008. Overview of BioCreative II gene normalization. *Genome Biology*, 9(Suppl 2).

Paul Nulty and Fintan Costello. 2010. UCD-PN: Selecting general paraphrases using conditional probability. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval-2)*, Uppsala, Sweden.

Diarmuid Ó Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, OH.

Joseph Reisinger and Raymond Mooney. 2010. A mixture model with sharing for lexical semantics. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, Cambridge,MA.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent Dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, College Park, MD.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1–2):159–216.

Karen Spärck Jones. 1964. *Synonymy and Semantic Classification*. Ph.D. thesis, University of Cambridge.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations us-

ing syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, Uppsala, Sweden.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Hanna Wallach, David Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *Proceedings of NIPS-09*, Vancouver, BC.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-09)*, Paris, France.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics (COLING-00)*, Saarbrücken, Germany.

Lexical Co-occurrence, Statistical Significance, and Word Association

Dipak L. Chaudhari
Computer Science and Engg.
IIT Bombay
dipakc@cse.iitb.ac.in

Om P. Damani
Computer Science and Engg.
IIT Bombay
damani@cse.iitb.ac.in

Srivatsan Laxman
Microsoft Research India
Bangalore
slaxman@microsoft.com

Abstract

Lexical co-occurrence is an important cue for detecting word associations. We propose a new measure of word association based on a new notion of statistical significance for lexical co-occurrences. Existing measures typically rely on global unigram frequencies to determine expected co-occurrence counts. Instead, we focus only on documents that contain both terms (of a candidate word-pair) and ask if the distribution of the observed spans of the word-pair resembles that under a random null model. This would imply that the words in the pair are not related strongly enough for one word to influence placement of the other. However, if the words are found to occur closer together than explainable by the null model, then we hypothesize a more direct association between the words. Through extensive empirical evaluation on most of the publicly available benchmark data sets, we show the advantages of our measure over existing co-occurrence measures.

1 Introduction

Lexical co-occurrence is an important indicator of word association and this has motivated several co-occurrence[1] measures for word association like PMI (Church and Hanks, 1989), LLR (Dunning, 1993), Dice (Dice, 1945), and CWCD (Washtell and Markert, 2009). In this paper, we present a new measure of word association based on a new notion of statistical significance for lexical co-occurrences. In general, a lexical co-occurrence could refer to a pair of words that co-occur in a large number of documents; or it could refer to a pair of words that, although co-occur only in a small number of documents, occur close to each other within those documents. We formalize these ideas and construct a significance test that allows us to detect different kinds of co-occurrences within a single unified framework (a feature which is absent in current measures for co-occurrence). Another distinguishing feature of our measure is that it is based solely on the co-occurrence counts in the documents containing both words of the pair, unlike all existing measures which also take global unigram frequencies in account.

We need a null hypothesis that can account for an observed co-occurrence as a pure chance event and this in-turn requires a corpus generation model. Documents in a corpus can be assumed to be generated independent of each other. Existing co-occurrence measures further assume that each document is drawn from a multinomial distribution based on global unigram frequencies. The main concern with such a null model is the overbearing influence of the unigram frequencies on the detection of word associations. For example, the association between *anomochilidae* (dwarf pipe snakes) and *snake* could go undetected in our wikipedia corpus, since less than 0.1% of the pages containing *snake* also contained *anomochilidae*. Also, under current models, the expected *span*[2] of a word pair is very sensitive to the associated unigram frequencies: the expected span of a word pair composed of low frequency unigrams is much larger than that with high frequency unigrams. This is contrary to how word associa-

[1] We use the term co-occurrence to refer to a pair of words that co-occur in a document with an arbitrary number of intervening words.

[2] The *span* of an occurrence of a word-pair is the 'unsigned distance' between the positions of the corresponding word occurrences.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1058–1068,
Edinburgh, Scotland, UK, July 27–31, 2011.

tions appear in language, where semantic relationships manifest with small inter-word distances irrespective of the underlying unigram distributions.

Based on these considerations we employ a null model that represents each document as a bag of words [3]. A random permutation of the associated bag of words gives a linear representation for the document. Under this null model, the locations of an unrelated pair of words will likely be randomly distributed in the documents in which they co-occur. If the observed span distribution of a word-pair resembles that under the (random permutation) null model, then the relation between the words is not strong enough for one word to influence the placement of the other. However, if the words are found to occur closer together than explainable by our null model, then we hypothesize a more direct association between the words. Therefore, this null model detects biases in span distributions of word-pairs while being agnostic to variations in global unigram frequencies.

In this paper, we propose a new measure of word association based on the statistical significance of the observed span distribution of a word-pair. We perform extensive experiments on all the publicly available benchmark data sets[4] and compare our measure against other popular co-occurrence measures. Our experiments demonstrate the advantages of our measure over all the competing measures. The ranked list of word associations output by our measure has the best correlation with the corresponding gold-standard in three (out of seven) data sets in our experiments, while remaining in the top three in other four datasets. While different measures perform best on different data sets, our measure outperforms other measures by being consistently either the best measure or very close to the best measure on all the data sets. The average deviation of our measure's correlation with the gold-standard from the best measure's correlation with the gold-standard (average taken across all the datasets) is 0.02, which is the least average deviation among all the measures, the next best deviations being 0.04 and 0.06.

The paper is organized as follows. We present our notion of statistical significance of span distribution in Section 2. Algorithm for computing the proposed word association measure is described in Section 3. We discuss related work in Section 4. Performance evaluation is presented in Section 5 followed with conclusions in Section 6.

2 Lexically significant co-occurrences

Evidence for significant lexical co-occurrences can be gathered at two levels in the data – document-level and corpus-level. First, at the document level, we may find that for a given word-pair, a surprisingly high proportion of its occurrences *within* a document have smaller spans than they would have by random chance. Second, at the corpus-level, we may find a pair of words appearing closer-than-random in multiple documents in the corpus. We now describe how to combine both kinds of evidence to decide whether the nearby occurrences of a word-pair are statistically significant or not.

Let the *frequency* f of a word-pair α in a document D, be the maximum number of *non-overlapped occurrences* of α in D. A set of occurrences of a word-pair is said to be non-overlapped if the words corresponding to one occurrence from the set do not appear in-between the words corresponding to any other occurrence from the set.

Let $\widehat{f}^x$ denote the maximum number of non-overlapped occurrences of α in D with span less than a given threshold x. We refer to $\widehat{f}^x$ as the *span-constrained frequency* of α in D. Note that $\widehat{f}^x$ cannot exceed f.

2.1 Document-level significant co-occurrence

To assess the statistical significance of the word-pair α we ask if the span-constrained frequency $\widehat{f}^x$ (of α) is more than what we would expect in a document of size ℓ containing f 'random' occurrences of α. Our intuition is that if two words are associated in some way, they will often appear close to each other in the document and so the distribution of the spans will typically exhibit a bias toward values less than a suitably chosen threshold x.

[3] There can be many ways to associate a bag of words with a document. Details of this association are not important for us, except that the bag of words provides some kind of quantitative summary of the words within the document.

[4] We exclude very small data sets of 80 word pairs or less. Sizes of the seven datasets we used range from 351 word-pairs to 83,713 word-pairs.

Definition 1 *Consider the null hypothesis that the linear representation of a document is generated by choosing a random permutation of the bag of words associated with the document. Let ℓ be the length of the document and f denote the frequency of a word-pair in the document. For a given a span threshold x, we define $\pi_x(\widehat{f^x}, f, \ell)$ as the probability under the null that the word-pair will appear in the document with a span-constrained frequency of* at least $\widehat{f^x}$.

Observe that $\pi_x(0, f, \ell) = 1$ for any $x > 0$; also, for $x \geq \ell$ we have $\pi_x(f, f, \ell) = 1$ (i.e. all f occurrences will always have span less than x for $x \geq \ell$). However, for typical values of x (i.e. for $x \ll \ell$) the probability $\pi_x(\widehat{f^x}, f, \ell)$ decreases with increasing $\widehat{f^x}$. For example, consider a document of length 400 with 4 non-overlapped occurrences of α. The probabilities of observing at least 4, 3, 2, 1 and 0 occurrences of α within a span of 20 words are 0.007, 0.09, 0.41, 0.83, and 1.0 respectively. Since $\pi_{20}(3, 4, 400) = 0.09$, even if 3 of the 4 occurrences of α have span less than 20 words, there is 9% chance that the occurrences were a consequence of a random event. As a result, if we desired a confidence-level of at least 95%, we would have to declare observed co-occurrences of α as *insignificant*.

Given an ϵ ($0 < \epsilon < 1$) and a span threshold x (≥ 0) the document D is said to *support* the hypothesis "α is an ϵ-significant word-pair within the document" if we have $[\pi_x(\widehat{f^x}, f, \ell) < \epsilon]$. We refer to ϵ as the *document-level* evidence of the lexical co-occurrence of α.

2.2 Corpus-level significant co-occurrence

We now describe how to aggregate evidence for lexical significance by considering the occurrence of α across multiple documents in the corpus. Let $\{D_1, \ldots, D_K\}$ denote the set of K documents (from out of the entire corpus) that contain at least one occurrence of α. Let ℓ_i be the length of D_i, f_i be the frequency of α in D_i, and, $\widehat{f_i^x}$ be the *span-constrained frequency* of α in D_i. Define indicator variables z_i, $i = 1, \ldots, K$ as:

$$z_i = \begin{cases} 1 & \text{if } \pi_x(\widehat{f_i^x}, f_i, \ell_i) < \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

As discussed previously, z_i indicates whether "α is an ϵ-significant word-pair within the document D_i." Note that we view $\widehat{f_i^x}$ as the only random quantity here, with x fixed by the user, and ℓ_i and f_i fixed given the document D_i and word-pair α. Let $Z = \sum_{i=1}^{K} z_i$; Z models the number of documents (out of K) that support the hypothesis "α is an ϵ-significant word-pair." The expected value of Z is given by

$$\begin{aligned} E(Z) &= \sum_{i=1}^{K} E(z_i) \\ &= \sum_{i=1}^{K} \pi_x(g_{\epsilon,x}(f_i, \ell_i), f_i, \ell_i) \quad (2) \end{aligned}$$

where $g_{\epsilon,x}(f_i, \ell_i)$ is given by *Definition 2* below.

Definition 2 *Given a document of length ℓ in which a word-pair has a frequency of f, and given a span threshold x, we define $g_{\epsilon,x}(f, \ell)$ as the smallest r for which the inequality $[\pi_x(r, f, \ell) < \epsilon]$ holds.*

Note that $g_{\epsilon,x}(f, \ell)$ is well-defined since $\pi_x(r, f, \ell)$ is non-increasing with respect to r. For the example given earlier, $g_{0.2,20}(4, 400) = 3$ and $g_{0.05,20}(4, 400) = 4$. Since each document in the corpus is assumed to be generated independently, z_i's are independent random variables and we can bound the deviation of the observed value of Z from its expectation using Hoeffding's Inequality – for any $t > 0$, we have

$$\begin{aligned} P[Z \geq E(Z) + Kt] &\leq \exp(-2Kt^2) \\ &= \delta \quad (3) \end{aligned}$$

Recall that Z models the number of documents supporting the hypothesis "α is an ϵ-significant word-pair."). Thus, the upper-bound δ ($= \exp(-2Kt^2)$), $0 < \delta < 1$ denotes the upper-bound on the probability that just due to random chance, more than $(E(Z) + Kt)$ documents out of K will support the hypothesis "α is an ϵ-significant word-pair." We call δ the *corpus-level* evidence of the lexical co-occurrence α. For example, in our corpus, the word-pair (*canyon, landscape*) occurs in $K = 416$ documents. For $\epsilon = 0.1$, we have ϵ-significant occurrences in $Z = 33$ documents (out of 416) , while $E(Z) = 14.34$. Suppose we want to be 99% sure that the occurrences of (*canyon, landscape*) in the 33 documents were a consequence of non-random phenomena. Let $\delta = 1 - 0.99 = 0.01$. By setting

word-1	*word-2*		
	$(0.1, 0.1)$	$(0.1, 0.4)$	$(0.4, 0.1)$
algae	green	mold	pool
amuse	entertain	clown	amaze
damn	hell	mad	bad
rat	dirty	ugly	disease
sedative	drug	narcotic	calm
topping	chocolate	flavour	caramel
umbrella	rain	dry	shade
unknown	known	dark	secret
worm	insect	dirt	fishing
wrap	cover	seal	bandage

Table 1: Examples of word-pairs from *Florida* dataset having statistically significant co-occurrences in the wikipedia corpus for different (ϵ, δ) combinations under a span constraint of 20 words.

$t = \sqrt{\ln \delta/(-2K)} = 0.07$, we get $E(Z) + Kt = 43.46$. Only if Z was 44 or more, there would be less than 1% chance of that being a random phenomena. Thus, we cannot be 99% sure that the observed co-occurrences in the 33 documents are non-random. Hence, our test declares (*canyon, landscape*) as *insignificant* at $\epsilon = 0.1, \delta = 0.01$. We now summarize our significance test in the definition below.

Definition 3 (Significant lexical co-occurrence) *Consider a word-pair α and a set of K documents containing at least one occurrence each of α. Fix a span threshold of x (> 0), a document-level evidence of ϵ ($0 < \epsilon < 1$) and a corpus-level evidence of δ ($0 < \delta < 1$). Let Z denote the number of documents (out of K) that support the hypothesis "α is ϵ-significant within the document." The word-pair α is said to be (ϵ, δ)-significant if we have $[Z \geq E(Z) + Kt]$, where $t = \sqrt{\log \delta/(-2K)}$ and $E(Z)$ is given by Eq. (2). The ratio $[Z/(E(Z) + Kt)]$ is called the* Co-occurrence Significance Ratio (CSR) *for α.*

2.3 Discussion

The significance test of *Definition 3* gathers both document-level and corpus-level evidence from data in calibrated amounts. Prescribing ϵ fixes the strength of the document-level hypothesis in our test, while, δ, controls the extent of corpus-level evidence we need to declare a word-pair as significant. A small δ demands that there must be multiple documents in the corpus, each of which, individually have some evidence of relatedness for the pair of words.

By running the significance test with different values of ϵ and δ, the CSR test can be used to detect different types of lexically significant co-occurrences. For example, the strongest lexical co-occurrences would have both strong document-level evidence (low ϵ) as well as high corpus-level evidence (low δ). Informally, these would represent pairs of words that appear multiple times with small spans within a document, in many documents, and in-practice, we find that multi-word expressions or pairs of words separated by stop words tend to dominate this type. On the other hand, a higher ϵ would represent word-pairs that appear relatively farther apart within a document, or a higher δ would represent word-pairs that appear together in relatively fewer documents. Note that to detect co-occurrences that exclusively correspond to (say) low ϵ and high δ, we would have to run the test with low ϵ and high δ, and then remove word-pairs that were also found significant at low ϵ and low δ.

In Table 1, we present some examples of different types of co-occurrences. The table lists word-pairs that were found to be statistically significant for different choices of (ϵ, δ). Note that a word-pair is reported under $(\epsilon = 0.1, \delta = 0.4)$ or $(\epsilon = 0.4, \delta = 0.1)$ only if it was not also found significant under other two parameter settings. The strongest correlations are the word-pairs corresponding to $(\epsilon = 0.1, \delta = 0.1)$ e.g., *algae-green*, *rat-dirty* and *worm-insect*. Different sets of weaker co-occurrences are detected depending on whether we relaxed δ or ϵ. For example, *algae-mold* is significant at a higher δ, while *algae-pool* is significant for higher ϵ.

The semantic notion of word association is an abstract concept and different kinds of associations (with potentially different statistical characterizations) may be preferred by human judges in different situations. While in Section 5, we discuss in detail various datasets used, the evaluation methodology, and the performance of CSR across datasets, we wish to point out here that in 3 out of 5 cross-validation runs for *wordsim* dataset, the best performing CSR parameters were $x = 50w$, $\epsilon = 0.1$ and $\delta = 0.9$, while in 3 out of 5 runs for *Minnesota* dataset, the best performing CSR parameters were $x = 20w$, $\epsilon = 0.3$ and $\delta = 0.5$. This gives us some indication that different kinds of word associations were preferred in different data sets.

3 Computing Co-occurrence Significance Ratio(CSR)

There are three main steps for computing CSR and the pseudocodes for these are listed in Procedures 1, 2 & 3. Of these, the first two can be run offline since they do not depend on the text corpus. They need to be run only once, after which CSR can be computed for any word-pair on any given corpus of documents. We describe these steps in the subsections below.

3.1 Computing histogram $hist_{f,\ell,x}(\cdot)$

The first step is to compute a histogram for the span-constrained frequency, $\widehat{f^x}$, of a word-pair whose frequency is f in a document of length ℓ, given a chosen span threshold of x (under our null model).

Definition 4 *Given a document of length ℓ and a span threshold of x, we define $hist_{f,\ell,x}(\widehat{f^x})$ as the number of ways to embed f non-overlapped occurrences of a word-pair in the document such that exactly $\widehat{f^x}$ occurrences have span less than x.*

Procedure 1 $ComputeHist(f, \ell, x)$ – Offline

Input f - number of non-overlapped occurrences; ℓ - document length; x - span threshold
Computes $hist_{f,\ell,x}[\cdot]$ as per *Definition 4*

```
1:  Initialize hist_{f,ℓ,x}[f̂^x] ← 0 for f̂^x = 0, ..., f
2:  if f > ℓ then
3:      return
4:  if f = 0 then
5:      hist_{f,ℓ,x}[0] ← 1
6:      return
7:  for i ← 1 to (ℓ − 1) do
8:      for j ← (i + 1) to ℓ do
9:          hist_{f−1,ℓ−j,x} ← ComputeHist(f − 1, ℓ − j, x)
10:         for k ← 0 to f − 1 do
11:             if (j − i) < x then
12:                 hist_{f,ℓ,x}[k + 1] ← hist_{f,ℓ,x}[k + 1]
                                          + hist_{f−1,ℓ−j,x}[k]
13:             else
14:                 hist_{f,ℓ,x}[k] ← hist_{f,ℓ,x}[k] + hist_{f−1,ℓ−j,x}[k]
```

Procedure 1 lists the pseudocode for computing the histogram $hist_{f,\ell,x}$. The main steps involve selecting a start and end position for embedding the very first occurrence (lines 7-8) and then recursively calling $ComputeHist(\cdot,\cdot,\cdot)$ (line 9). The i-loop selects a start position for the first occurrence of the word-pair, and the j-loop selects the end position. The recursion step now computes the number of ways to embed the remaining $(f-1)$ non-overlapped occurrences in the remaining $(\ell - j)$ positions. Once we have $hist_{f-1,\ell-j}$, we check whether the occurrence introduced at positions (i, j) will contribute to the $\widehat{f^x}$ count. If $(j - i) < x$, whenever there are k span-constrained occurrences in positions $(j + 1)$ to ℓ, there will be $(k + 1)$ span-constrained occurrences in positions 1 to ℓ. Thus, we increment $hist_{f,\ell}[k + 1]$ by the quantity $hist_{f-1,\ell-j}[k]$ (lines 10-12). However, if $(j - i) > x$, there is no contribution to the span-constrained frequency from the (i, j) occurrence, and so we increment $hist_{f,\ell}[k]$ by the quantity $hist_{f-1,\ell-j}[k]$ (lines 10-11, 13-14). Finally, we note that in our implementation we use memorization to avoid redundant recursive calls.

3.2 Computing $\pi_x(\cdot, f, \ell)$ distribution

Procedure 2 $ComputePiDist(f, \ell, x)$ – Offline

Input f - number of non-overlapped occurrences; ℓ - document length; x - span threshold
Computes Distribution $\pi_x[f, \ell, \cdot]$ as per *Definition 1* and $g_{\epsilon,x}[f, \ell]$ as per *Definition 2*

```
1: N[f, ℓ, x] = Σ_{k=0}^{f} hist_{f,ℓ,x}[k]
2: for f̂^x ← 0 to f do
3:     N_x[f̂^x, f, ℓ] ← Σ_{k=f̂^x}^{f} hist_{f,ℓ,x}[k]
4:     π_x[f̂^x, f, ℓ] ← N_x[f̂^x, f, ℓ] / N[f, ℓ, x]
5: g_{ϵ,x}[f, ℓ] ← min{r | π_x[r, f, ℓ] < ϵ}
```

The second offline step is computation of the $\pi_x(\cdot, f, \ell)$ distribution. We store the number of ways of embedding f non-overlapped occurrences of a word-pair in a document of length ℓ in the array $N[f, \ell]$. Similarly, the array $N_x[\widehat{f^x}, f, \ell]$ stores the number of ways of embedding f non-overlapped occurrences of the word-pair in a document of length ℓ, such that at least $\widehat{f^x}$ of the f occurrences have span less than x. To compute $N[f, \ell, x]$ and $N_x[\widehat{f^x}, f, \ell]$, we need the histogram $hist_{f,\ell,x}[\cdot]$ which is the output of *Procedure 1*. *Procedure 2* lists the pseudocode for computing $\pi_x(\widehat{f^x}, f, \ell)$ from $N(f, \ell)$ and $N_x(\widehat{f^x}, f, \ell)$ given $hist_{f,\ell}$ from *Procedure 1* (For the sake of readability the pseudocode does not describe some optimizations that we used in our implementation).

The *Procedure 1* is exponential in f and ℓ but it does not depend on the data corpus. Hence, we can run the Procedures 1 and 2 off-line, and publish the $\pi_x[]$ and $g_{\epsilon,x}[]$ tables for various x, $\widehat{f^x}$, f and

ℓ. Using these tables[5], anyone wishing to compute CSR needs to only run Procedure 3.

3.3 Computing CSR for a given word-pair

Procedure 3 $ComputeCSR(\alpha, \epsilon, \delta, x)$

Input α - word-pair; ϵ - document-level evidence; δ - corpus-level evidence; x - span threshold; Corpus of documents
Computes $CSR(\alpha)$ - Co-occurrence Significance Ratio (CSR) for α as per *Definition 3*

1: $\mathcal{D} \leftarrow \{D_1, \ldots, D_K\}$ // Set of documents from the corpus that each contain at least one occurrence of α.
2: $t \leftarrow \sqrt{\log \delta/(-2K)}$
3: $Z \leftarrow 0$ and $Z_E \leftarrow 0$
4: **for** $i \leftarrow 1$ to K **do**
5: $\quad \ell_i$ = Length of D_i
6: $\quad f_i$ = Frequency of α in D_i
7: $\quad \widehat{f}_i^x$ = Span-constrained frequency of α in D_i
8: $\quad$ **if** $\pi_x[\widehat{f}_i^x, f_i, \ell_i] < \epsilon$ **then**
9: $\quad\quad z_i \leftarrow 1$
10: $\quad$ **else**
11: $\quad\quad z_i \leftarrow 0$
12: $\quad Z \leftarrow Z + z_i$
13: $\quad Z_E \leftarrow Z_E + \pi_x[g_\epsilon[f_i, \ell_i, x], f_i, \ell_i]$
14: $CSR(\alpha) = Z/(Z_E + Kt)$

Procedure 3 implements the significance test given in *Definition 3* and requires that the $\pi_x[]$ and $g_{\epsilon,x}[]$ tables have already been computed offline.

The first step is to determine the subset $\mathcal{D}$ of documents containing the given word-pair (line 1). Then we compute t based on δ and K (the size of $\mathcal{D}$) (line 2). Next we determine how many of the K documents support the hypothesis "α is ϵ-significant within the document" (lines 3-12). The expected number of documents supporting the hypothesis is accumulated in Z_E (line 13). CSR is then computed as the ratio of Z to $(Z_E + Kt)$ (line 14).

3.4 Run-time overhead

The computation of Co-occurrence Significance Ratio (CSR) as given in Definition 3 might appear more complex than the simple formulae for other co-occurrence measures given in Table 2. However, bulk of the complexity in calculating CSR lies in the one-time (data independent) off-line computation of the $\pi_x[]$ and $g_{\epsilon,x}[]$ tables. Once these tables are published, the cost of comparing CSR for a given word pair is comparable to the cost of computing any other (spanned) measure in Table 2. The main data-dependent computations for a spanned measure are in determining span-constrained frequencies; all other steps are simple arithmetic operations or memory lookups. To illustrate this, Procedure 4 gives details of computing PMI. The comparison of Procedures 3 and 4 shows their almost parallel structures. The main overhead in these procedures is incurred in line 7, where span-constrained frequencies in a given document are computed.

Procedure 4 $ComputePMI(a, b)$

Input (x, y) - word pair;
Computes PMI (Table 2) for (x, y).

1: let $\mathcal{D} = \{D_1, \ldots, D_K\}$ // set of documents containing at least one occurrence of α.
2: N = total number of words in corpus
3: (f_x,f_y) = unigram frequencies of x, y in corpus
4: (p_x,p_y) = (f_x/N,f_y/N)
5: $\widehat{f} = 0$
6: **for** $i \leftarrow 1$ to K **do**
7: $\quad \widehat{f}_i$ = span-constrained frequency of α in D_i
8: $\quad \widehat{f} = \widehat{f} + \widehat{f}_i$
9: $\hat{p}_{x,y} = \widehat{f}/N$
10: $PMI = log(\frac{\hat{p}_{x,y}}{p_x p_y})$

4 Related Work

Existing word association measures can be divided into three broad categories: (i) *Co-occurrence measures* that rely on co-occurrence frequencies of both words in a corpus in addition to the individual unigram frequencies (Table 2), (ii) *Distributional similarity-based measures* that characterize a word by the distribution of other words around it (Agirre et al., 2009; Bollegala et al., 2007; Chen et al., 2006; Wandmacher et al., 2008), and (iii) *Knowledge-based measures* that use knowledge-sources like thesauri, semantic networks, or taxonomies (Milne and Witten, 2008; Hughes and Ramage, 2007; Gabrilovich and Markovitch, 2007; Yeh et al., 2009; Strube and Ponzetto, 2006; Finkelstein et al., 2002; Liberman and Markovitch, 2009).

In this paper, we focus on comparison with other co-occurrence measures. These measures are used in several domains like ecology, psychology, medicine, and language processing. Table 2 lists several measures chosen from all these domains. Except Ochiai (Ochiai, 1957; Janson and Vegelius, 1981) and the recently introduced

[5] http://www.cse.iitb.ac.in/~damani/papers/EMNLP11/resources.html

Method	Formula
CSR (this work)	$Z/(E(Z)+Kt)$
CWCD (Washtell and Markert, 2009)	$\frac{\hat{f}(x,y)}{p(x)} \frac{1/max(p(x),p(y))}{M}$
Dice (Dice, 1945)	$\frac{2\hat{f}(x,y)}{f(x)+f(y)}$
LLR (Dunning, 1993)	$\sum_{\substack{x' \in \{x,\neg x\} \\ y' \in \{y,\neg y\}}} p(x',y') log \frac{p(x',y')}{p(x')p(y')}$
Jaccard (Jaccard, 1912)	$\frac{\hat{f}(x,y)}{f(x)+f(y)-\hat{f}(x,y)}$
Ochiai (Janson and Vegelius, 1981)	$\frac{\hat{f}(x,y)}{\sqrt{f(x)f(y)}}$
Pearson's χ^2 test	$\sum_{\substack{x' \in \{x,\neg x\} \\ y' \in \{y,\neg y\}}} \frac{\left(\hat{f}(x',y')-E\hat{f}(x',y')\right)^2}{E\hat{f}(x',y')}$
PMI (Church and Hanks, 1989)	$log \frac{p(x,y)}{p(x)p(y)}$
SCI (Washtell and Markert, 2009)	$\frac{p(x,y)}{p(x)\sqrt{p(y)}}$
T-test	$\frac{\hat{f}(x,y)-E\hat{f}(x,y)}{\sqrt{\hat{f}(x,y)\left(1-\frac{\hat{f}(x,y)}{N}\right)}}$

N	Total number of tokens in the corpus
$f(x), f(y)$	unigram frequencies of x, y in the corpus
$p(x), p(y)$	$f(x)/N, f(y)/N$
$\hat{f}(x,y)$	Span-constrained (x,y) word pair frequency in corpus
$\hat{p}(x,y)$	$\hat{f}(x,y)/N$
M	Harmonic mean of the spans of $\hat{f}(x,y)$ occurrences
$E\hat{f}(x,y)$	Expected value of $\hat{f}(x,y)$

Table 2: Co-occurrence measures.

CWCD[6] (Washtell and Markert, 2009) all other measures are well-known in the NLP community (Pecina and Schlesinger, 2006). Our results show that Ochiai and Chi-Square have almost identical performance, differing only in 3rd decimal digits. Rankings produced by Chi-square is almost monotonic with respect to the rankings produced by Ochiai. This is because, for most word pairs (x,y), $[f(x) \ll N]$, $[f(y) \ll N]$, $[f(x,y) \ll f(x)]$, and $[f(x,y) \ll f(y)]$. Therefore three of the four terms in the Chi-square summation become zero[7] and the fourth term approximates to the square of Ochiai. Similarly Jaccard and Dice coincide. While presenting our experimental results, we report these pairs of measures together.

[6]CWCD was reported in (Washtell and Markert, 2009) as the best performing variant among the so-called windowless (or spanless) measures. In our experiments, we implemented windowed (spanned) version of the CWCD measure.

[7]For example, $\hat{f}(x,\neg y) - Ef(x,\neg y) = f(x) - N \times pf(x) \times pf(\neg y) = f(x) - \frac{1}{N} \times f(x) \times f(\neg y) = f(x) - \frac{1}{N} \times f(x) \times N = 0$.

Aspect	Data Set	No. of Respondents	No. of Word Pairs	No. of Filtered Word Pairs
Semantic relatedness	wordsim (Finkelstein et al., 2002)	16	353	351
Free-Association	Edinburg (Kiss et al., 1973)	100	325,588	83,713
	Florida (Nelson et al., 1980)	5,019	65,523	59,852
	Goldfarb-Halpern (Goldfarb and Halpern, 1984)	316	410	384
	Kent (Kent and Rosanoff, 1910)	1,000	14,576	14,086
	Minnesota (Russell and Jenkins, 1954)	1,007	10,447	9,649
	White-Abrams (White and Abrams, 2004)	440	745	652

Table 3: Characteristics of data sets used.

5 Performance Evaluation

Two main aspects of word association studied in literature are: a) *semantic relatedness*, and b) *free association*. *Semantic relatedness* encompasses many different relationships between words, like synonymy, meronymy, antonymy, and functional association (Budanitsky and Hirst, 2006). *Free association* refers to the first response-words that come to mind when presented with a stimulus. (ESSLLI, 2008). We experiment with all the publicly available datasets that come with gold standard judgement of these aspects, except the very small ones with less than 80 word-pairs[8].

5.1 Datasets

Details[9] of the datasets used in our experiments are listed in Table 3. Each data set comes with a gold-standard of human judgments - a ranked list of association scores for the word-pairs in the data set. The *wordsim* dataset was prepared by asking the subjects to estimate the relatedness of the word pairs on a

[8](MillerCharles (Miller and Charles, 1991), Rubenstein-Goodenough (Rubenstein and Goodenough, 1965) and TOEFL (Landauer and Dumais, 1997))

[9]We removed word-pairs containing multiword expressions. For data sets with more than 10,000 word-pairs, we filtered out pairs that contain stop words listed in (StopWordList, 2010). For Edinburg (size 275393 after previous filtering), we further filtered word-pairs where the response was supported by only one respondant. Original and filtered data sets are available at http://www.cse.iitb.ac.in/~damani/papers/EMNLP11/resources.html

	Edinburg (83,713)	Florida (59,852)	Kent (14,086)	Minnesota (9,649)	White-Abrams (652)	Goldfarb-Halpern (384)	wordsim (351)
CSR	**0.25**	**0.30**	0.42	0.31	**0.34**	0.10	0.63
CWCD	0.23	0.23	0.40	0.30	0.21	**0.19**	0.54
Dice (Jaccard)	0.20	0.27	**0.43**	**0.32**	0.21	0.09	0.59
LLR	0.20	0.26	0.40	0.29	0.18	0.03	0.51
Ochiai (χ^2)	0.24	**0.30**	**0.43**	0.31	0.29	0.08	0.62
PMI	0.22	0.25	0.36	0.26	0.22	0.11	**0.69**
SCI	0.24	0.27	0.38	0.27	0.23	0.06	0.37
TTest	0.17	0.23	0.37	0.26	0.17	-0.02	0.45

Table 4: Comparison of the average Spearman coefficients obtained across five cross-validation runs by different measures. The best performing measure for each data-set is shown in bold. All standard deviations for Edinburg and Florida were less than 0.01, for Kent and Minnesota were between 0.01 and 0.02, for White-Abrams were between 0.05 and 0.08, for Goldfarb-Halpern between 0.05 and 0.15 and for wordsim were between 0.02 and 0.15. Number of word-pairs in each dataset is shown in brackets against its name.

	Edinburg (83,713)	Florida (59,852)	Kent (14,086)	Minnesota (9,649)	White-Abrams (652)	Goldfarb-Halpern (384)	wordsim (351)	Worst Rank	Avg. Deviation	Worst Deviation
CSR	**0.00 (1)**	**0.00 (1)**	0.01 (3)	0.01 (2)	**0.00 (1)**	0.09 (3)	0.06 (2)	**3**	**0.02**	**0.09**
CWCD	0.02 (4)	0.07 (7)	0.03 (4)	0.02 (4)	0.13 (5)	**0.00 (1)**	0.15 (5)	7	0.06	0.15
Dice (Jaccard)	0.05 (6)	0.03 (3)	**0.00 (1)**	**0.00 (1)**	0.13 (5)	0.10 (4)	0.10 (4)	6	0.06	0.13
LLR	0.05 (6)	0.04 (5)	0.03 (4)	0.03 (5)	0.16 (7)	0.16 (7)	0.18 (6)	7	0.09	0.18
Ochiai (χ^2)	0.01 (2)	**0.00 (1)**	**0.00 (1)**	0.01 (2)	0.05 (2)	0.11 (5)	0.07 (3)	5	0.04	0.11
PMI	0.03 (5)	0.05 (6)	0.07 (8)	0.06 (7)	0.12 (4)	0.08 (2)	**0.00 (1)**	8	0.06	0.12
SCI	0.01 (2)	0.03 (3)	0.05 (6)	0.05 (6)	0.11 (3)	0.13 (6)	0.32 (8)	8	0.10	0.32
TTest	0.08 (8)	0.07 (7)	0.06 (7)	0.06 (7)	0.17 (8)	0.21 (8)	0.24 (7)	8	0.13	0.24

Table 5: Comparison of deviations from the best performing measure on each data set. Number of word-pairs in each dataset is shown in brackets against its name. Figures in brackets against the deviation values denote the ranks of the measures in the corresponding data sets.

scale from 0 to 10 (Finkelstein et al., 2002). The methodology for collecting *free association* data is explained at (ESSLLI, 2008): The degree of free association between a stimulus (S) and response (R) is the percentage of respondents who respond R as the first response when presented with stimulus S.

These datasets are of varying size, and they were constructed at different point in time, in different geographies. This allows us to compare different measures comprehensively under varying range of circumstances. To the best of our knowledge, no previous work has reported such a detailed comparison of co-occurrence measures.

5.2 Resources Used

We use the Wikipedia (Wikipedia, April 2008) corpus with 2.7 million articles (total of 1.24 Gigawords). We did no pre-processing - no lemmatization or function-word removal. When counting document size (in words), punctuations were ignored. Documents larger than 1500 words were partitioned such that each part was at most 1500 words[10]. We indexed the corpus using Lucene search engine library and used Lucene APIs to obtain various statistics and documents containing given word-pairs.

5.3 Methodology

Each measure listed in Table 2 produces a ranked list of association scores for the word-pairs in a data set. We evaluate each measure by the Spearman's rank correlation between the ranking produced by the measure and the gold-standard ranking.

The span threshold (or window-width) x is a user-defined parameter in all measures. In addition, CSR has the parameters ϵ and δ. For any measure, the ranking of word-pairs will likely change with chang-

[10]While this limit can be raised using heavier computing resources, we believe that partitioning documents of sizes greater than 1500 words was reasonable (especially since typical span values we used were less than 50, much less than 1500).

Method	Resource	wordsim			Esslli (272)
		wordsim (353)	sim (203)	rel (252)	
PMI	Wikipedia	0.69	0.72	0.68	0.32
Ochiai (χ^2)	Wikipedia	0.62	0.68	0.62	**0.44**
Significance Ratio (CSR)	Wikipedia	0.63	0.70	0.64	0.43
Latent Semantic Analysis (Wandmacher et al., 2008)	Newspaper corpus	-	-	-	0.38
Graph Traversal (WN30g) (Agirre et al., 2009))	Wordnet	0.66	0.72	0.56	-
Bag of Words based Distributional Similarity (BoW) (Agirre et al., 2009))	Web corpus	0.65	0.70	0.62	-
Context Window based Distributional Similarity (CW) (Agirre et al., 2009))	Web corpus	0.60	**0.77**	0.46	-
Hyperlink Graph (Milne and Witten, 2008)	Wikipedia hyperlinks graph	0.69	-	-	-
Random Graph Walk (Hughes and Ramage, 2007)	WordNet	0.55	-	-	-
Explicit Semantic Analysis (Gabrilovich and Markovitch, 2007) (reimplemented in (Yeh et al., 2009))	Wikipedia concepts	**0.75** **(0.71)**	-	-	-
Normalized Path-length (lch) (Strube and Ponzetto, 2006)	Wikipedia category tree	0.55	-	-	-
Thesarus based (Jarmasz, 2003)	Roget's thesaurus	0.55	-	-	-
Latent Semantic Analysis (Finkelstein et al., 2002)	Web corpus	0.56	-	-	-

Table 6: Comparison of co-occurrence based measures with knowledge-based and distributional similarity based measures. These other measures have not been applied to the free association datasets shown in Table 3. Data for missing entries is not available. Note that *sim* and *rel* are subsets of *wordsim* dataset. Number of word-pairs in each dataset is shown in brackets against its name.

ing parameter values. Hence we follow the standard methodology of fixing parameters through cross validation. Specifically, we partition the data into five folds, four of which are used for training and one hold-out fold is used for testing. For each measure, the parameter values that achieve best correlation with human judgments on 4 training folds are used to predict on the 1 hold-out testing fold. This experiment is repeated 5 times for different training and test folds. The average rank correlation obtained by each measure over 5 cross-validation runs is reported for each dataset. We varied ϵ and δ between 0.01 and 0.90 and x between 5 and 50 words.

5.4 Results

For each measure and for each data set, the average correlation over the 5 cross-validation runs is reported in Table 4. The corresponding standard deviations are mentioned in the table's caption. The best performing measure in each case is highlighted in bold. While different measures performed best on different data sets, the results in Table 4 shows that CSR performs consistently well across all data sets. In all data sets the correlation for CSR was always either the best or close to the best.

As expected, our results are statistically more significant for the larger data sets, compared to the smaller ones. The standard deviations of the results are small for two largest data sets (less than 0.01 for Edinburg and Florida), gradually increasing (less than 0.02 for Kent and Minnesota), and becoming high (upto .15) for the three smallest datasets.

Although, among all measures, CSR has the best average correlation over all datasets, taking average of correlations across widely different dataset is not a meaningful way to decide on which measure to use. Ideally one would like to access an oracle to learn which measure will perform best on a particular unseen application dataset. Short of such an oracle, if one were to pick a fixed measure a-priori, then one would like to know how much worse off one is compared to the best measure for that dataset.

To compare different measures from this perspective, we compute the deviation of the correlation for each measure from the correlation of the best measure for each data set. These deviations are reported in Table 5, along with the corresponding ranks. The average deviation of CSR over all the data sets is 0.02, which is the least among all the measures, the next two being 0.04 and 0.06. CSR also has the least worst-deviation among all measures. Also, CSR is never ranked worse than 3 in any of the data sets. This is also the smallest worst-rank among all measures. Based on these results, we infer that CSR is overall the best performing co-occurrence based word association measure.

While the focus of our work is on the co-occurrence measures, for completeness, we present all the known results for knowledge and distributional similarity-based measures on the datasets un-

der consideration in Table 6. Note that in (Agirre et al., 2009), the *wordsim* data set was partitioned into two sets, namely *sim* and *rel*, and in *Esslli* shared task (ESSLLI, 2008), a 272 word pair subset of the *Edinburgh* dataset was chosen. To facilitate comparison, in addition to CSR, we also present results for PMI and Ochiai (Chi-Square) which are the best performing co-occurrence measures on *wordsim*, and *Esslli* datasets. For co-occurrence-based measures, we used 5-fold cross validation, which is inapplicable for parameterless measures. Results show that co-occurrence-based measures compare well with other resource-heavy measures.

6 Conclusions

In this paper, we introduced a new measure called CSR for word-association based on statistical significance of lexical co-occurrences. Our measure, while being agnostic to global unigram frequencies, detects skews in span distributions of word-pairs in documents containing both words. We carried out extensive evaluation on several benchmark datasets. Our experiments demonstrate the advantages of our measure over all the competing measures.

Acknowledgments

This work was supported in part by the Ministry of Human Resources Development, Government of India and by the Tata Research Development and Design Center (TRDDC). We thank Mr. Justin Washtell (University of Leeds) for providing us with various datasets.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL-HLT*.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Measuring semantic similarity between words using web search engines. In *WWW*, pages 757–766.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguists*, 32(1):13–47.

Hsin-Hsi Chen, Ming-Shun Lin, and Yu-Chuan Wei. 2006. Novel association measures using web search with double checking. In *ACL*.

Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information and lexicography. In *ACL*, pages 76–83.

L. R. Dice. 1945. Measures of the amount of ecological association between species. *Ecology*, 26:297–302.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

ESSLLI. 2008. *Free association* task at lexical semantics workshop esslli 2008. `http://wordspace.collocations.de/doku.php/workshop:esslli:task`.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*.

Robert Goldfarb and Harvey Halpern. 1984. Word association responses in normal adult subjects. *Journal of Psycholinguistic Research*, 13(1):37–55.

T Hughes and D Ramage. 2007. Lexical semantic relatedness with random graph walks. In *EMNLP*.

P. Jaccard. 1912. The distribution of the flora of the alpine zone. *New Phytologist*, 11:37–50.

Svante Janson and Jan Vegelius. 1981. Measures of ecological association. *Oecologia*, 49:371–376.

Mario Jarmasz. 2003. Rogets thesaurus as a lexical resource for natural language processing. Technical report, University of Ottowa.

G. Kent and A. Rosanoff. 1910. A study of association in insanity. *American Journal of Insanity*, pages 317–390.

G. Kiss, C. Armstrong, R. Milroy, and J. Piper. 1973. An associative thesaurus of english and its computer analysis. In *The Computer and Literary Studies*, pages 379–382. Edinburgh University Press.

T. Landauer and S. Dumais. 1997. The latent semantic analysis theory of acquisition, induction, and representation of knowledge. In *Psychological Review*, volume 104/2, pages 211–240.

Sonya Liberman and Shaul Markovitch. 2009. Compact hierarchical explicit semantic representation. In *Proceedings of the IJCAI 2009 Workshop on User-Contributed Knowledge and Artificial Intelligence: An Evolving Synergy (WikiAI09)*, Pasadena, CA, July.

G.A. Miller and W.G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *ACL*.

D. Nelson, C. McEvoy, J. Walling, and J. Wheeler. 1980. The university of south florida homograph norms. *Behaviour Research Methods and Instrumentation*, 12:16–37.

A Ochiai. 1957. Zoogeografical studies on the soleoid fishes found in japan and its neighbouring regions-ii. *Bulletin of the Japanese Society of Scientific Fisheries*, 22.

Pavel Pecina and Pavel Schlesinger. 2006. Combining association measures for collocation extraction. In *ACL*.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, October.

W.A. Russell and J.J. Jenkins. 1954. The complete minnesota norms for responses to 100 words from the kent-rosanoff word association test. Technical report, Office of Naval Research and University of Minnesota.

StopWordList. 2010. `http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words`. *The Information Retrieval Group, University of Glasgow*. Accessed: November 15, 2010.

Michael Strube and Simone Paolo Ponzetto. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *AAAI*, pages 1419–1424.

T. Wandmacher, E. Ovchinnikova, and T. Alexandrov. 2008. Does latent semantic analysis reflect human associations? In *European Summer School in Logic, Language and Information (ESSLLI'08)*.

Justin Washtell and Katja Markert. 2009. A comparison of windowless and window-based computational association measures as predictors of syntagmatic human associations. In *EMNLP*, pages 628–637.

Katherine K. White and Lise Abrams. 2004. Free associations and dominance ratings of homophones for young and older adults. *Behavior Research Methods, Instruments, & Computers*, 36(3):408–420.

Wikipedia. April 2008. `http://www.wikipedia.org`.

Eric Yeh, Daniel Ramage, Chris Manning, Eneko Agirre, and Aitor Soroa. 2009. Wikiwalk: Random walks on wikipedia for semantic relatedness. In *ACL workshop "TextGraphs-4: Graph-based Methods for Natural Language Processing"*.

Learning the Information Status of Noun Phrases in Spoken Dialogues

Altaf Rahman and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{altaf,vince}@hlt.utdallas.edu

Abstract

An entity in a dialogue may be old, new, or mediated/inferrable with respect to the hearer's beliefs. Knowing the information status of the entities participating in a dialogue can therefore facilitate its interpretation. We address the under-investigated problem of automatically determining the information status of discourse entities. Specifically, we extend Nissim's (2006) machine learning approach to information-status determination with lexical and structured features, and exploit learned knowledge of the information status of each discourse entity for coreference resolution. Experimental results on a set of Switchboard dialogues reveal that (1) incorporating our proposed features into Nissim's feature set enables our system to achieve state-of-the-art performance on information-status classification, and (2) the resulting information can be used to improve the performance of learning-based coreference resolvers.

1 Introduction

Information status is not a term unfamiliar to researchers working on discourse processing problems. It describes the extent to which a discourse entity, which is typically a noun phrase (NP), is *available* to the hearer given the speaker's assumptions about the hearer's beliefs. According to Nissim et al. (2004), a discourse entity can be *new*, *old*, or *mediated*. Informally, a discourse entity is (1) *old* to the hearer if it is known to the hearer and has previously been referred to in the dialogue, (2) *new* if it is unknown to her and has not been previously referred to; and (3) *mediated* if it is newly mentioned in the dialogue but she can infer its identity from a previously-mentioned entity. Information status is a subject that has received a lot of attention in theoretical linguistics (Halliday, 1976; Prince, 1981; Hajičová, 1984; Vallduví, 1992; Steedman, 2000).

Knowing the information status of discourse entities can potentially benefit many NLP applications. One such task is anaphora resolution. While there is general belief that definite descriptions are mostly anaphoric, Vieira and Poesio (2000) empirically show that only 30% of these NPs are anaphoric. Without being able to determine whether an NP is anaphoric, an anaphora resolver will attempt to resolve every NP, potentially damaging its precision. Since *new* entities are by definition new to the hearer and therefore cannot refer to a previously-introduced NP, knowledge of information status could be used to improve anaphora resolution.

Despite the potential usefulness of information status in NLP tasks, there has been little work on *learning* the information status of discourse entities. To investigate the plausibility of learning information status, Nissim et al. (2004) annotate a set of Switchboard dialogues with such information[1], and subsequently present a rule-based approach and a learning-based approach to acquiring such knowledge from the manual annotations (Nissim, 2006).

Our goals in this paper are two-fold. First, we describe a learning approach to the under-studied problem of determining the information status of discourse entities that extends Nissim's (2006) feature set with two novel types of features: lexical features and structured features based on syntactic parse trees. Second, we employ the automatically

[1]These and other linguistic annotations on the Switchboard dialogues were later released by the LDC as part of the NXT corpus, which is described in detail in Calhoun et al. (2010).

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1069–1080,
Edinburgh, Scotland, UK, July 27–31, 2011.

acquired knowledge of information status for coreference resolution. Experimental results on Nissim et al.'s (2004) corpus of Switchboard dialogues show that (1) adding our linguistic features to Nissim's feature set enables our system to outperform her system by 8.1% in F-measure, and (2) learned knowledge of information status can be used to improve coreference resolvers by 1.1–2.6% in F-measure.

The rest of this paper is organized as follows. We first illustrate with examples the concepts of *new*, *old*, and *mediated* entities. Then, we describe the dataset and the feature set that Nissim (2006) used in her approach. After that, we introduce our lexical and structured features. Finally, we evaluate the determination of information status as a standalone task and in the context of coreference resolution.

2 Old, New, and Mediated Entities

Since the concepts of *old*, *new*, and *mediated* entities are not widely known to researchers working outside the area of discourse processing, in this section we will explain them in more detail.

The terms *old* and *new* information have meant a variety of things over the years (Allerton, 1978; Prince, 1981; Horn, 1986). Since we use Nissim et al.'s (2004) corpus for training and evaluation, the definitions of these concepts we present here are those that Nissim et al. used to annotate their corpus. According to Nissim et al., their definitions are built upon Prince's (1981), and the categorization into *old*, *new*, and *mediated* entities resemble those of Strube (1998) and Eckert and Strube (2001).

Old. As mentioned before, an entity is *old* if it is both known to the hearer and has been mentioned in the conversation. More precisely, an entity is *old* if (1) it is coreferential with an entity introduced earlier, (2) it is a generic pronoun, or (3) it is a personal pronoun referring to the dialogue participants. To exemplify, consider the following sentences.

(1) I was angry that he destroyed **my** tent.
(2) **You** cannot leave until the test is over.

In Example 1, *my* is an *old* entity because it is coreferent with *I*. In Example 2, *You* is an *old* entity because it is a generic pronoun.

Mediated. An entity is *mediated* if it has not been previously introduced in the conversation, but can be inferred from already-mentioned entities or is generally known to the hearer. More specifically, an entity is *mediated* if (1) it is a generally known entity (e.g., *the Earth*, *China*, and most proper names), (2) it is a bound pronoun, or (3) it is an instance of *bridging* (i.e., an entity that is inferrable from a related entity mentioned earlier in the dialogue). As an example, consider the following sentences.

(3a) He passed by the door of Mary's house and saw that **the door** was painted purple.
(3b) He passed by Mary's house and saw that **the door** was painted purple.

In Example 3a, by the time the hearer processes the second occurrence of *the door*, she has already had a mental entity corresponding to *the door* (after processing the first occurrence). As a result, the second occurrence of *the door* is an *old* entity. In Example 3b, on the other hand, the hearer is not assumed to have any mental representation of the door in question, but she can infer that the door she saw was part of Mary's house. Hence, this occurrence of *the door* is a *mediated* entity. In general, an entity that is related to an earlier entity via a part-whole relation or a set-subset relation is *mediated*.

New. An entity is *new* if it has not been introduced in the dialogue and the hearer cannot infer it from previously mentioned entities.

In case more than one class is appropriate for a given entity, Nissim et al. employ additional tie-breaking rules. Suppose, for instance, that we have two occurrences of *China* in a dialogue. The second occurrence can be labeled as *old* (because it is coreferential with an earlier entity) or *mediated* (because it is a generally known entity). According to Nissim et al.'s rules, the entity will be labeled as *old*.

3 Dataset

We employ Nissim et al.'s (2004) dataset, which comprises 147 Switchboard dialogues. A total of 68,992 NPs are annotated with information status: 51.2% of them are labeled as *old*, 34.5% as *mediated* (henceforth *med*), and 14.3% as *new*. Nissim (2006) randomly split the instances created from these NPs into a training set (for classifier training), a development set (for feature development), and an evaluation set (for testing). Hence, the NPs from the same

	Training	Test
old	31358 (51.7%)	3931 (47.4%)
med	20778 (34.2%)	3036 (36.6%)
new	8567 (14.1%)	1322 (16.0%)
total	60703 (100%)	8289 (100%)

Table 1: Information status distribution of NPs.

document may be split across different sets.

Unlike Nissim (2006), we partition the 147 dialogues (rather than the instances) into a training set (117 dialogues) and a test set (30 dialogues). In other words, we do *not* randomize the instances, as we believe that it represents an unrealistic evaluation setting, for the following reasons. First, in practice, the test dialogues may not be available until test time. Second, we may want to examine how a system performs on a given dialogue. Finally, randomizing the instances does not allow us to apply learned knowledge of information status to coreference resolution, which needs to be performed for each dialogue. The information status distribution of the NPs in the training and test sets are shown in Table 1.

4 Baseline System

In this section, we describe our baseline system, which adopts a machine learning approach to determining the information status of a discourse entity.

Building SVM classifiers for information-status determination. We employ the support vector machine (SVM) learner as implemented in the SVM^{light} package (Joachims, 1999) to train three binary classifiers, one for predicting each of the three possible classes (i.e., *new*, *old*, and *med*), using a linear kernel in combination with the *one-versus-all* training scheme.[2] Each training instance represents a single NP and consists of the seven morpho-syntactic features that Nissim (2006) used in her learning-based approach (see Table 2 for an overview). Following Nissim, we extract the NPs directly from the gold-standard annotations, but the features are computed entirely automatically.

[2]SVM was chosen because it provides the option to employ kernels. The reason why we train three binary classifiers rather than just one multi-class classifier (using SVM^{multiclass}) is that SVM^{multiclass} does not permit the use of a non-linear kernel, which we will need to incorporate structured features later on.

Feature	Values
full prev mention	numeric
mention time	{first,second,more}
partial prev mention	{yes,no,NA}
determiner	{bare,def,dem,indef,poss,NA}
NP type	{pronoun,common,proper,other}
NP length	numeric
grammatical role	{subject,subjpass,pp,other}

Table 2: Nissim's feature set.

The seven features are all intuitively useful for determining information status. For instance, if an NP, NP$_k$, and a discourse entity that appears before it have the same string (full prev mention), then NP$_k$ is likely to be an *old* entity. Mention time is the categorical version of full prev mention and therefore serves to detect *old* entities. Partial prev mention is useful for detecting mediated entities, especially those that have a set-subset relation with a preceding entity. For instance, *your dogs* would be considered a partial previous mention of *my dogs* or *my three dogs*. The value "NA" stands for "not applicable", and is used for pronouns. Determiners and NP type are likely to be helpful for all three categories. For instance, indefinite NPs and pronouns are likely to be *new* and *old*, respectively. The "NP length" feature is motivated by the observation that *old* entities tend to contain less lexical materials than *new* entities. For instance, subsequent references to *Barack Obama* may simply be *Obama*.

Applying the classifiers. To determine the information status of an NP in a test dialogue, we create an instance for it as during training and present it independently to the three binary SVM classifiers, each of which returns a real value representing the signed distance of the instance from the hyperplane. We assign the instance to the class that is associated with the most positive classification value.

5 Our Features

We propose to extend Nissim's (2006) feature set with two types of features.

5.1 Lexical Features

As discussed, an entity should be labeled as *med* if it has not been introduced in the dialogue but is gener-

ally known to a human. Whether an entity is "generally known" may be easily determined by a human but not by a machine, since world knowledge is involved in the decision process. In particular, Nissim's feature set does not contain any features that encode the notion of a "generally known" entity.

Hence, it would be desirable to augment Nissim's feature set with features that indicate whether an entity is generally known or not. One way to do this is to (1) create a list of generally known entities, and then (2) create a binary feature that has the value *True* if and only if the entity under consideration appears in this list. The question, then, is: how can we obtain the list of generally known entities? We may manually assemble this list, but this could be a labor-intensive task. As a result, we propose to *acquire* this kind of world knowledge automatically from annotated data.

Specifically, we augment Nissim's feature set with the set of *unigrams* that appear in the training data. Given a training/test instance (i.e., discourse entity), we compute the values of its unigram features (henceforth *lexical features*) as follows. For each unigram, we check if it appears in the string representing the discourse entity. If so, its feature value is 1; otherwise, its value is 0. For instance, if the entity is *the red hat*, then all of its lexical features except *the*, *red*, and *hat* will have a value of 0.

It should perhaps not be too difficult to see why these lexical features are useful for the information-status classifier: these features enable the SVM learner to determine the extent to which a unigram correlates with each class. For instance, from the annotated data, the learner will learn that any instance of *China* cannot be labeled as *new*, and the decision of whether it should be an *old* entity or a *med* entity depends on whether it is coreferential with a previously-mentioned entity. Hence, the use of lexical features allows the learner to implicitly acquire some world knowledge.

We believe that lexicalization is an important step towards building high-performance text-processing systems. In fact, lexicalized models have demonstrated their effectiveness in other areas of language processing, such as syntactic and semantic parsing. While lexicalized models may be less portable to new genres and domains than their unlexicalized counterparts, we believe that this issue can be handled via domain adaptation techniques and should not be a reason against lexicalization.

5.2 Structured Features

In Nissim's (2006) feature set, there are a couple of features that capture NP-internal information, such as determiner, NP length, and NP type. However, there is only one feature that captures the syntactic context of an NP, grammatical role, which is computed based on the parse tree in which the NP resides. This is arguably a very shallow representation of its syntactic context. We hypothesize that we can train more accurate information-status classifiers if we have access to a richer representation of syntactic context. This motivates us to employ syntactic parse trees *directly* as features.

Before describing how this can be done, recall that in a traditional learning setting, the feature set employed by an off-the-shelf learning algorithm typically consists of *flat* features (i.e., features whose values are discrete- or real-valued, as the ones described in the previous section). Advanced machine learning algorithms such as SVMs, on the other hand, have enabled the use of *structured* features (i.e., features whose values are structures such as parse trees), owing to their ability to employ *kernels* to efficiently compute the similarity between two potentially complex structures.

Perhaps the main advantage of employing structured features is *simplicity*. To understand this advantage, consider learning in a setting where we can only employ flat features. If we want to use information from a parse tree as features in this setting, we will need to design heuristics to extract the desired parse-based features from parse trees. For certain tasks, designing a good set of heuristics can be time-consuming and sometimes difficult. On the other hand, SVMs enable a parse tree to be employed directly as a structured feature, obviating the need to design such heuristics.

Given two parse trees (as features), we compute their similarity using a convolution tree kernel (Collins and Duffy, 2001), which efficiently enumerates the number of common substructures in the two trees via dynamic programming. Note, however, that while we want to use a parse tree directly as a feature, we do *not* want to use the *entire* parse tree as a feature. Specifically, while using the entire parse

tree enables a richer representation of the syntactic context than using a *partial* parse tree, the increased complexity of the tree also makes it more difficult for the SVM learner to make generalizations.

To strike a better balance between having a rich representation of the context and improving the learner's ability to generalize, we extract a substructure from a parse tree and use it as the value of the structured feature of an instance. Specifically, given an instance corresponding to discourse entity e, we extract the substructure from the parse tree containing e as follows. Let $n(e)$ be the root of the subtree that spans all and only the words in e, and let $Parent(n(e))$ be its immediate parent node. We (1) take the subtree rooted at $Parent(n(e))$, (2) replace each leaf node in this subtree with a node labeled X, (3) replace the subtree rooted at $n(e)$ with a leaf node labeled Y, and (4) use the subtree rooted at $Parent(n(e))$ as the structured feature for the instance corresponding to e. Intuitively, the first three steps aim to provide generalizations by simplifying the tree. For instance, step (1) allows us to focus on using a small window as the context. Steps (2) and (3) help generalization by ignoring the words within e and its context. Note that using two labels, X and Y, enables the kernel to distinguish the discourse entity under consideration from its context within this substructure. In addition, we simply use a single node (Y) to represent the discourse entity, since any NP-internal information has presumably been captured by the flat features. We compute these structured features using hand-annotated parse trees.

While structured features have been employed for a multitude of tasks in syntax, semantics, and information extraction such as syntactic parsing (e.g., Collins (2002)), semantic parsing (e.g., Moschitti (2004)), named entity recognition (e.g., Cumby and Roth (2003), and relation extraction (e.g., Zelenko et al. (2003)), the same is not true for discourse processing tasks. We hope that our use of structured features for information-status classification can promote their use in discourse processing.

5.3 Combining Kernels

Recall that the flat features are computed using a linear kernel, while the structured features are computed using a tree kernel. If we want our learner to make use of more than one of these types of features, we need to employ a *composite* kernel to combine them. Specifically, we define and employ the following composite kernel:

$$K_c(F_1, F_2) = K_1(F_1, F_2) + K_2(F_1, F_2),$$

where F_1 and F_2 are the full set of features that represent the two entities under consideration, and K_1 and K_2 are the kernels we are combining. To ensure that both kernels contribute equally to the composite kernel, we normalize the values they return to the range [0,1].

6 Evaluation

Next, we evaluate the effectiveness of our features in improving information-status classification.

6.1 Results and Discussion

Results of four information-status classification systems are shown in Table 3. Under Original Nissim, we have the results copied verbatim from Nissim's (2006) paper. Baseline is the aforementioned baseline system, which is trained using Nissim's feature set. Baseline+Lexical is the system trained using Nissim's feature set augmented with lexical features. Finally, Baseline+Both is the system trained using Nissim's feature set augmented with both lexical and structured features. For each system, we show the recall (R), precision (P), and F-measure (F) of each of the three classes: *old*, *new*, and *med*. Before we describe the results, two points deserve mention. First, as noted earlier, Nissim partitioned the dialogues into training and test folds in a different way than us. In particular, Original Nissim and the remaining three systems were not evaluated on the same set of test instances. Hence, the Original Nissim results are not directly comparable with those of the other systems. We show them here just to provide another point of reference. Second, the results of the remaining three systems were obtained by aggregating the results of three binary SVM classifiers, as described earlier.

Comparing Baseline and Baseline+Lexical, we see that augmenting Nissim's feature set with lexical features improves the F-measure scores on all three classes. In particular, the F-measure and recall for *med* rise considerably by 3.0 and 7.8, respectively. This provides indirect empirical support for our earlier hypothesis that the *med* class can benefit from

	Original Nissim			Baseline			Baseline+Lexical			Baseline+Both		
	R	P	F	R	P	F	R	P	F	R	P	F
old	91.5	94.1	92.8	91.2	85.8	88.5	88.7	91.7	90.2	93.0	95.2	**94.1**
med	87.6	68.1	76.6	84.7	62.7	72.1	92.5	63.2	75.1	89.1	70.9	**79.0**
new	22.3	56.3	32.0	30.2	66.4	41.5	32.1	68.3	43.7	34.4	71.5	**46.5**
Accuracy	79.5			74.1			76.3			**82.2**		

Table 3: Per-class performance of four information-status classifiers.

the shallow world knowledge that these lexical features help to "extract" from annotated data.

Comparing Baseline+Lexical and Baseline+Both, we see that the addition of structured features enables a further boost to performance: F-measure increases by 2.8–3.9 for the three classes. These results substantiate our hypothesis that employing a richer representation of syntactic context is beneficial to information-status classification.

Comparing Baseline and Baseline+Both, we see that F-measure improves considerably by 5–6.9 for the three classes. Overall, these results provide suggestive evidence that both types of features are effective at improving an information-status classifier that employs Nissim's features.

For further comparison, we show the classification accuracies of the four systems in the last row of Table 3. As we can see, adding lexical features to the baseline features improves accuracy by 2.2%, and adding structured features further improves accuracy by 5.9%. Our two types of features, when used in combination with Nissim's features, improve the baseline substantially by an accuracy of 8.1%.

Note that while our results and Original Nissim's are not directly comparable, the two systems are consistent in terms of the relative performance for the three classes: best for *old* and worst for *new*. The poor performance for *new* is largely a consequence of its low recall, which can in turn be attributed to its lower representation in the dataset. Since many *new* instances are misclassified, a natural question is: are these instances misclassified as *old* or *med*? Similar questions can be raised for *old* and *med*, despite their substantially higher recall values than *new*.

To answer these questions, we need to better understand the kind of errors made by our approach. Consequently, we show in Table 4 the confusion matrix generated from the test set for our best-performing information-status classifier, Baseline+Both. The rows and the columns correspond to the gold tags and the classifier tags, respectively. As we can see, these numbers seem to suggest the "in-between" nature of mediated entities: when an *old* or *new* entity is misclassified, it is typically misclassified as *med* (rows 1 and 3); however, when a *med* entity is misclassified, it is equally likely to be misclassified as *old* and *new* (row 2).

C→ G↓	old	med	new
old	3656	257	18
med	167	2706	163
new	17	850	455

Table 4: Confusion matrix for the Baseline+Both classifier. C=Classifier tag; G=Gold tag

These results are perhaps not surprisingly, since intuitively *med* entities bear some resemblance to both *old* and *new* entities. For instance, the similarity between *med* and *old* stems from the fact that different instances of the same entity (e.g., *China*) can receive one of these two labels, with the decision dependent on whether the entity was previously mentioned in the dialogue. On the other hand, *med* and *new* are similar in that it may sometimes be difficult even for a human to determine whether certain entities should be labeled as *med* or *new*, since the decision depends on whether she believes these entities are *generally known* or not.

6.2 Relation to Anaphoricity Determination

Anaphoricity determination refers to the task of determining whether an NP is anaphoric or not, where an NP is considered anaphoric if it is part of a (non-singleton) coreference chain but is not the head of the chain (Ng and Cardie, 2002). In other words, an

	Anaphoricity			Baseline+Ana			Baseline+Lexical+Ana			Baseline+Both+Ana		
	R	P	F	R	P	F	R	P	F	R	P	F
old	91.4	86.6	88.9	91.3	87.3	89.3	90.8	91.7	91.3	92.8	94.9	93.9
med	84.3	63.1	72.2	84.9	64.1	73.1	92.3	64.7	76.1	88.7	71.1	78.9
new	30.8	66.4	42.1	31.1	66.9	42.5	32.9	68.7	44.5	34.1	71.7	46.2
Accuracy	74.7			75.1			77.6			82.0		

Table 5: Impact of knowledge of anaphoricity on the information-status classifiers.

NP is anaphoric if and only if it has an antecedent.

Given this definition, anaphoricity determination bears resemblance to information-status classification. For instance, an *old* entity is anaphoric, since it has been introduced earlier in the conversation and therefore have an antecedent. Similarly, a *new* or *med* entity is non-anaphoric, since the entity has not been previously introduced in the conversation and therefore cannot have an antecedent.

There has been a lot of recent work on anaphoricity determination (e.g., Bean and Riloff (1999), Uryupina (2003), Ng (2004), Denis and Baldridge (2007), Versley et al. (2008), Ng (2009), Zhou and Kong (2009)). Given the similarity between this task and information-status classification, a natural question is: will the anaphoricity features previously developed by coreference researchers be helpful for information-status classification? To answer this question, we (1) assemble a feature set composed of the 26 anaphoricity features previously used by Rahman and Ng (2009),[3] and then (2) repeat the experiments in Table 3, except that we augment the feature set used in each of these experiments with the anaphoricity features we assembled in step (1).

Results with the anaphoricity features are shown in Table 5. Under Anaphoricity, we have the results obtained using only the 29 anaphoricity features. As we can see, these results are comparable to those obtained using the Baseline features. Comparing each of Baseline+Ana and Baseline+Lexical+Ana with the corresponding experiments in Table 3, we see that the addition of anaphoricity features yields a mild performance improvement, which is consistent over all three classes. However, comparing the last column of the two tables, we can see that in the presence of the structured features, the anaphoricity features do not contribute positively to overall performance. Hence, in the coreference experiments in the next section, we will not employ anaphoricity features for information-status classification.

[3]These 26 features are derived from those employed by Ng and Cardie's (2002) anaphoricity determination system. See Footnote 2 of Rahman and Ng (2009) for details.

7 Application to Coreference Resolution

Since the significance of information-status classification stems in part from the potential benefits it brings to higher-level NLP applications, we determine whether our information-status classification systems can offer benefits to learning-based coreference resolution. Since the 147 information-status annotated dialogues are also coreference annotated, we use them in our coreference evaluation. To our knowledge, our work represents the first attempt to report coreference results on this dataset.

7.1 Coreference Models

While the so-called mention-pair coreference model has dominated coreference research for more than a decade since its appearance in the mid-1990s, a number of new coreference models have been proposed in recent years. To investigate whether these newer, presumably more sophisticated, coreference models can better exploit the automatically acquired information-status information, we will evaluate the usefulness of information-status information when used in combination with two different coreference models, the aforementioned mention-pair model and the recently-developed cluster-ranking model.

7.1.1 Mention-Pair Model

The mention-pair (MP) model, proposed by Aone and Bennett (1995) and McCarthy and Lehnert (1995), is a classifier that determines whether two NPs are co-referring or not. Each instance $\boldsymbol{i}(\mathrm{NP}_j, \mathrm{NP}_k)$ corresponds to two NPs, NP_j and NP_k, and is represented by 39 features. Table 1 of Rahman and

Ng (2009) contains a detailed description of these features. Linguistically, they can be divided into four categories: string-matching, grammatical, semantic, and positional. They can also be categorized based on whether they are relational or not. Specifically, relational features capture the relationship between NP_j and NP_k, whereas non-relational features capture the linguistic property of one of these NPs.

We follow Soon et al.'s (2001) method for creating training instances. Specifically, we create (1) a positive instance for each anaphoric NP NP_k and its closest antecedent NP_j; and (2) a negative instance for NP_k paired with each of the intervening NPs, NP_{j+1}, NP_{j+2}, ..., NP_{k-1}. The classification associated with a training instance is either positive or negative, depending on whether the two NPs are coreferent. To train the MP model, we use the SVM learner from SVM^{light} (Joachims, 1999).[4]

After training, the classifier is used to identify an antecedent for an NP in a test text. Specifically, each NP, NP_k, is compared in turn to each preceding NP, NP_j, from right to left, and selects NP_j as its antecedent if the pair is classified as coreferent. The process terminates as soon as an antecedent is found for NP_k or the beginning of the text is reached.

Despite its popularity, the MP model has two major weaknesses. First, since each candidate antecedent for an NP to be resolved (henceforth an *active NP*) is considered independently of the others, this model only determines how good a candidate antecedent is relative to the active NP, but not how good a candidate antecedent is relative to other candidates. So, it fails to answer the critical question of which candidate antecedent is most probable. Second, it has limitations in its expressiveness: the information extracted from the two NPs alone may not be sufficient for making a coreference decision.

7.1.2 Cluster-Ranking Model

The cluster-ranking (CR) model, proposed by Rahman and Ng (2009), addresses the two weaknesses of the MP model by combining the strengths of the *entity-mention* model (e.g., Luo et al. (2004), Yang et al. (2008)) and the *mention-ranking* model (e.g., Denis and Baldridge (2008)). Specifically, the CR model ranks the preceding clusters for an active NP so that the highest-ranked cluster is the one to which the active NP should be linked. Employing a ranker addresses the first weakness, as a ranker allows all candidates to be compared *simultaneously*. Considering preceding clusters rather than antecedents as candidates addresses the second weakness, as *cluster-level* features (i.e., features that are defined over any subset of NPs in a preceding cluster) can be employed.

Since the CR model ranks preceding clusters, a training instance $\boldsymbol{i}(c_j, \text{NP}_k)$ represents a preceding cluster c_j and an anaphoric NP NP_k. Each instance consists of features that are computed based solely on NP_k as well as cluster-level features, which describe the relationship between c_j and NP_k. Motivated in part by Culotta et al. (2007), we create cluster-level features from the *relational* features in our feature set using four predicates: NONE, MOST-FALSE, MOST-TRUE, and ALL. Specifically, for each relational feature X, we first convert X into an equivalent set of binary-valued features if it is multi-valued. Then, for each resulting binary-valued feature X_b, we create four binary-valued cluster-level features: (1) NONE-X_b is true when X_b is false between NP_k and each NP in c_j; (2) MOST-FALSE-X_b is true when X_b is true between NP_k and less than half (but at least one) of the NPs in c_j; (3) MOST-TRUE-X_b is true when X_b is true between NP_k and at least half (but not all) of the NPs in c_j; and (4) ALL-X_b is true when X_b is true between NP_k and each NP in c_j.

We train a cluster ranker to jointly learn anaphoricity determination and coreference resolution using SVM^{light}'s ranker-learning algorithm. Specifically, for each NP, NP_k, we create a training instance between NP_k and *each* preceding cluster c_j using the features described above. Since we are learning a joint model, we need to provide the ranker with the option to start a new cluster by creating an additional training instance that contains features that solely describes NP_k. The rank value of a training instance $\boldsymbol{i}(c_j, \text{NP}_k)$ created for NP_k is the rank of c_j among the competing clusters. If NP_k is anaphoric, its rank is HIGH if NP_k belongs to c_j, and LOW otherwise. If NP_k is non-anaphoric, its rank is LOW unless it is the additional training instance described above, which has rank HIGH.

After training, the cluster ranker processes the NPs in a test text in a left-to-right manner. For each

[4] For this and subsequent uses of the SVM learner in our experiments, we set all parameters to their default values.

active NP, NP_k, we create test instances for it by pairing it with each of its preceding clusters. To allow for the possibility that NP_k is non-anaphoric, we create an additional test instance that contains features that solely describe the active NP (similar to what we did in the training step above). All these test instances are then presented to the ranker. If the additional test instance is assigned the highest rank value by the ranker, then NP_k is classified as non-anaphoric and will not be resolved. Otherwise, NP_k is linked to the cluster that has the highest rank.

7.2 Coreference Experiments

7.2.1 Experimental Setup

The training/test split we use in the coreference experiments is the same as that in the information-status experiments. Specifically, we use the training set to train both the information-status classifier and our coreference models, apply the information-status classifier to each discourse entity in the test set, and have the coreference models resolve all and only those NPs that are labeled as *old* by the information-status classifier. Our decision to allow the coreference models to resolve only the *old* entities is motivated by the fact that *med* and *new* entities have *not* been previously introduced in the conversation and therefore do not have antecedents. The NPs used by the coreference models are the same as those accessible to the information-status classifier.

We employ two scoring programs, B^3 (Bagga and Baldwin, 1998) and ϕ_3-CEAF (Luo, 2005), to score the output of a coreference model. Given a gold-standard (i.e., key) partition, KP, and a system-generated (i.e., response) partition, RP, B^3 computes the recall and precision of each NP and averages these values at the end. Specifically, for each NP, NP_j, B^3 first computes the number of NPs that appear in both KP_j and RP_j, the clusters containing NP_j in KP and RP, respectively, and then divides this number by $|KP_j|$ and $|RP_j|$ to obtain the recall and precision of NP_j, respectively. On the other hand, CEAF finds the best one-to-one alignment between the key clusters and the response clusters using the Kuhn-Munkres algorithm (Kuhn, 1955), where the weight of an edge connecting two clusters is equal to the number of NPs that appear in both clusters. Precision and recall are equal to the sum of the weights of the edges in the alignment divided by the total number of NPs in the response and the key, respectively.

7.2.2 Results and Discussion

As our baseline, we employ our coreference models to generate NP partitions on the test documents *without* using any knowledge of information status. Results, reported in terms of recall (R), precision (P), and F-measure (F) using B^3 and ϕ_3-CEAF, are shown in row 1 of Table 6.[5] As we can see, the baseline achieves B^3 F-measures of 69.2 (MP) and 74.5 (CR) and CEAF F-measures of 61.6 (MP) and 68.5 (CR). These results suggest that the CR model is stronger than the MP model, corroborating previous empirical findings (Rahman and Ng, 2009).

Next, we examine the impact of learned knowledge of information status on the performance of a coreference model. Since knowledge of information status enables a coreference model to focus on resolving only the *old* entities, we hypothesize that the resulting model will have a higher precision than one that does not employ such knowledge. An equally important question is: will the F-measure of the resulting model improve? Since we are employing knowledge of information status in a *pipeline* coreference architecture where information-status classification is performed prior to coreference resolution, errors made by the (upstream) information-status classifier may propagate to the (downstream) coreference system. Given this observation, we hypothesize that the answer to the aforementioned question depends in part on the accuracy of information-status classification. In particular, the higher the accuracy of information-status classification is, the more likely the F-measure of the downstream coreference model will improve. To test this hypothesis, we conduct experiments where we employ the knowledge provided by the three information-status classifiers which, as discussed earlier, perform at varying levels of accuracy — the first one using only Nissim's features, the second one using both lexical and Nissim's features, and the last one using Nissim's features in combination with lexical and parse-based features — for our coreference models.

[5] Since gold-standard NPs are used in our experiments, CEAF precision is always equal to CEAF recall. For brevity, we only report F-measure scores for CEAF in the table.

System	Mention-Pair Model				Cluster-Ranking Model			
	B^3			CEAF	B^3			CEAF
	R	P	F	F	R	P	F	F
No knowledge of information status	78.6	61.8	69.2	61.6	78.2	71.1	74.5	68.5
Nissim features only	73.4	67.3	70.2	62.1	73.6	77.4	75.4	69.7
Nissim+Lexical features	71.0	69.5	70.2	61.9	73.7	77.3	75.4	69.9
Nissim+Lexical+Parse features	74.1	66.8	70.3	62.3	77.3	74.0	75.6	71.1
Perfect information status	76.7	68.1	72.1	66.4	77.1	79.5	78.3	74.2

Table 6: B^3 and CEAF coreference results.

Results of the coreference models employing knowledge provided by the three information-status classifiers are shown in rows 2–4 of Table 6. As expected, B^3 precision increases in comparison to the baseline, regardless of the coreference model and the scoring program. In addition, employing knowledge of information status always improves coreference performance: F-measure scores increase by 1.0–1.1% (B^3) and 0.3–0.7% (CEAF) for the MP model, and by 0.9–1.1% (B^3) and 1.2–2.6% (CEAF) for the CR model. These results suggest that the three information-status classifiers have achieved the level of accuracy needed for the coreference models to improve. On the other hand, it is somewhat surprising that the three information-status classifiers have yielded coreference systems that perform at essentially the same level of performance.

To understand why better information-status classification results do not necessarily yield better coreference performance, we take a closer look at the results of the coreference resolver employing Nissim's features (henceforth NISSIM) and the resolver employing our Nissim+Lexical+Parse features (henceforth FULL-FEATURE). Among the *old* entities that were correctly classified using our features and incorrectly classified by Nissim's features, we found that the precision of the FULL-FEATURE system suffered (since in many cases the coreference models identified wrong antecedents for these *old* entities) whereas the NISSIM system remained unaffected (since the entities were misclassified and would not be resolved by the models). In addition, although many *med* and *new* entities were correctly classified using our features and incorrectly classified (as *old*) using Nissim's features, we found that in many cases no antecedents were identified for these misclassified entities and hence the precision of the NISSIM system was not adversely affected.

Finally, we investigate whether our coreference system could be improved if it had access to perfect knowledge of information status (taken directly from the gold-standard annotations). This experiment will allow us to determine whether the usefulness of knowledge of information status for coreference resolution is limited by the accuracy in computing such knowledge. Results are shown in the last row of Table 6. As we can see, using perfect information-status knowledge yields a coreference system that improves those that employs automatically acquired information-status knowledge by 1.8–4.1% (MP) and 2.7–3.1% (CR) in F-measure. This indicates that the accuracy in computing such knowledge does play a role in determining its usefulness for coreference resolution.

8 Conclusions

We examined the problem of automatically determining the information status of discourse entities in spoken dialogues. In particular, we augmented Nissim's feature set with two types of features: lexical features, which capture in a shallow manner world knowledge implicitly encoded in the annotated data; and syntactic parse trees, which provide a richer representation of the syntactic context in which a discourse entity appears than grammatical roles. Results on 147 Switchboard dialogues demonstrated the effectiveness of these features: we obtained a significant improvement of 8.1% in accuracy over a information-status classifier trained on Nissim's feature set. In addition, we evaluated information-status classification in the context of coreference resolution, and showed that automatically acquired knowledge of information status can be profitably used to improve coreference systems.

Acknowledgments

We thank the three reviewers for their invaluable comments on an earlier draft of the paper. This work was supported in part by NSF Grant IIS-0812261.

References

David J. Allerton. 1978. The notion of 'givenness' and its relations to presupposition and to theme. *Lingua*, 44:133–168.

Chinatsu Aone and Scott William Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 122–129.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at the First International Conference on Language Resources and Evaluation*, pages 563–566.

David Bean and Ellen Riloff. 1999. Corpus-based identification of non-anaphoric noun phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 373–380.

Sasha Calhoun, Jean Carletta, Jason Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The NXT-format Switchboard corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language Resources and Evaluation*, 44(4):387–419.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 489–496.

Aron Culotta, Michael Wick, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 81–88.

Chad Cumby and Dan Roth. 2003. On kernel methods for relational learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 107–114.

Pascal Denis and Jason Baldridge. 2007. Global, joint determination of anaphoricity and coreference resolution using integer programming. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 236–243.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669.

Miriam Eckert and Michael Strube. 2001. Dialogue acts, synchronising units and anaphora resolution. *Journal of Semantics*, 17(1):51–89.

Eva Hajičová. 1984. Topic and focus. In *Contributions to Functional Syntax, Semantics, and Language Comprehension (LLSEE 16)*, pages 189–202. John Benjamins, Amsterdam.

Michael A. K. Halliday. 1976. Notes on transitivity and theme in English. *Journal of Linguistics*, 3(2):199–244.

Laurence R. Horn. 1986. Presupposition, theme, and variations. In A. Farley et al., editor, *Papers from the Parasession on Pragmatics and Grammatical Theory at the 22nd Regional Meeting*, pages 168–192. CLS.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44–56. MIT Press.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(83).

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 135–142.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Joseph McCarthy and Wendy Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pages 1050–1055.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 335–342.

Vincent Ng and Claire Cardie. 2002. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 730–736.

Vincent Ng. 2004. Learning noun phrase anaphoricity to improve conference resolution: Issues in representation and optimization. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 151–158.

Vincent Ng. 2009. Graph-cut-based anaphoricity determination for coreference resolution. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 575–583.

Malvina Nissim, Shipra Dingare, Jean Carletta, and Mark Steedman. 2004. An annotation scheme for information status in dialogue. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*.

Malvina Nissim. 2006. Learning information status of discourse entities. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 94–102.

Ellen Prince. 1981. Toward a taxonomy of given-new information. In P. Cole, editor, *Radical Pragmatics*, pages 223–255. New York, N.Y.: Academic Press.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Michael Strube. 1998. Never look back: An alternative to centering. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 1251–1257.

Olga Uryupina. 2003. High-precision identification of discourse new and unique noun phrases. In *Proceedings of the ACL Student Research Workshop*, pages 80–86.

Enric Vallduví. 1992. *The Informational Component*. Garland, New York.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008a. Coreference systems based on kernels methods. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 961–968.

Renata Vieira and Massimo Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, and Sheng Li. 2008. An entity-mention model for coreference resolution with inductive logic programming. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 843–851.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

GuoDong Zhou and Fang Kong. 2009. Global learning of noun phrase anaphoricity in coreference resolution via label propagation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 978–986.

Harnessing WordNet Senses for Supervised Sentiment Classification

Balamurali A R[1,2] **Aditya Joshi**[2] **Pushpak Bhattacharyya**[2]
[1] IITB-Monash Research Academy, IIT Bombay
[2]Dept. of Computer Science and Engineering, IIT Bombay
Mumbai, India - 400076
{balamurali,adityaj,pb}@cse.iitb.ac.in

Abstract

Traditional approaches to sentiment classification rely on lexical features, syntax-based features or a combination of the two. We propose semantic features using word senses for a supervised document-level sentiment classifier. To highlight the benefit of sense-based features, we compare word-based representation of documents with a sense-based representation where WordNet senses of the words are used as features. In addition, we highlight the benefit of senses by presenting a part-of-speech-wise effect on sentiment classification. Finally, we show that even if a WSD engine disambiguates between a limited set of words in a document, a sentiment classifier still performs better than what it does in absence of sense annotation. Since word senses used as features show promise, we also examine the possibility of using similarity metrics defined on WordNet to address the problem of not finding a sense in the training corpus. We perform experiments using three popular similarity metrics to mitigate the effect of unknown synsets in a test corpus by replacing them with *similar* synsets from the training corpus. The results show promising improvement with respect to the baseline.

1 Introduction

Sentiment Analysis (SA) is the task of prediction of opinion in text. Sentiment classification deals with tagging text as positive, negative or neutral from the perspective of the speaker/writer with respect to a topic. In this work, we follow the definition of Pang et al. (2002) & Turney (2002) and consider a binary classification task for output labels as positive and negative.

Traditional supervised approaches for SA have explored lexeme and syntax-level units as features. Approaches using lexeme-based features use bag-of-words (Pang and Lee, 2008) or identify the roles of different *parts-of-speech (POS)* like adjectives (Pang et al., 2002; Whitelaw et al., 2005). Approaches using syntax-based features construct parse trees (Matsumoto et al., 2005) or use text parsers to model valence shifters (Kennedy and Inkpen, 2006).

Our work explores incorporation of semantics in a supervised sentiment classifier. We use the synsets in Wordnet as the feature space to represent word senses. Thus, a document consisting of words gets mapped to a document consisting of corresponding word senses. Harnessing WordNet senses as features helps us address two issues:

1. Impact of WordNet sense-based features on the performance of supervised SA
2. Use of WordNet similarity metrics to solve the problem of features unseen in the training corpus

The first points deals with evaluating sense-based features against word-based features. The second issue that we address is in fact an opportunity to improve the performance of SA that opens up because of the choice of sense space. Since sense-based features prove to generate superior sentiment classifiers, we get an opportunity to mitigate unknown

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1081–1091,
Edinburgh, Scotland, UK, July 27–31, 2011.

synsets in the test corpus by replacing them with known synsets in the training corpus. Note that such replacement is not possible if word-based representation were used as it is not feasible to make such a large number of similarity comparisons.

We use the corpus by Ye et al. (2009) that consists of travel domain reviews marked as positive or negative at the document level. Our experiments on studying the impact of Wordnet sense-based features deal with variants of this corpus manually or automatically annotated with senses. Besides showing the overall impact, we perform a POS-wise analysis of the benefit to SA. In addition, we compare the effect of varying training samples on a sentiment classifier developed using word based features and sense based features. Through empirical evidence, we also show that disambiguating some words in a document also provides a better accuracy as compared to not disambiguating any words. These four sets of experiments highlight our hypothesis that WordNet senses are better features as compared to words.

Wordnet sense-based space allows us to mitigate unknown features in the test corpus. Our synset replacement algorithm uses Wordnet similarity-based metrics which replace an unknown synset in the test corpus with the closest approximation in the training corpus. Our results show that such a replacement benefits the performance of SA.

The roadmap for the rest of the paper is as follows: Existing related work in SA and the differentiating aspects of our work are explained in section 2 Section 3 describes the sense-based features that we use for this work. We explain the similarity-based replacement technique using WordNet synsets in section 4. Our experiments have been described in section 5. In section 6, we present our results and related discussions. Section 7 analyzes some of the causes for erroneous classification. Finally, section 8 concludes the paper and points to future work.

2 Related Work

This work studies the benefit of a word sense-based feature space to supervised sentiment classification. However, a word sense-based feature space is feasible subject to verification of the hypothesis that sentiment and word senses are related. Towards this, Wiebe and Mihalcea (2006) conduct a study on human annotation of 354 words senses with polarity and report a high inter-annotator agreement. The work in sentiment analysis using sense-based features, including ours, assumes this hypothesis that *sense decides the sentiment*.

The novelty of our work lies in the following. Firstly our approach is distinctly. Akkaya et al. (2009) and Martn-Wanton et al. (2010) report performance of rule-based sentiment classification using word senses. Instead of a rule-based implementation, We used supervised learning. The supervised nature of our approach renders lexical resources unnecessary as used in Martn-Wanton et al. (2010). Rentoumi et al. (2009) suggest using word senses to detect sentence level polarity of news headlines. The authors use graph similarity to detect polarity of senses. To predict sentence level polarity, a HMM is trained on word sense and POS as the observation. The authors report that word senses particularly help understanding metaphors in these sentences. Our work differs in terms of the corpus and document sizes in addition to generating a general purpose classifier.

Another supervised approach of creating an emotional intensity classifier using concepts as features has been reported by Carrillo de Albornoz et al. (2010). This work is different based on the feature space used. The concepts used for the purpose are limited to affective classes. This restricts the size of the feature space to a limited set of labels. As opposed to this, we construct feature vectors that map to a larger sense-based space. In order to do so, we use synset offsets as representation of sense-based features.

Akkaya et al. (2009), Martn-Wanton et al. (2010) and Carrillo de Albornoz et al. (2010) perform sentiment classification of individual sentences. However, we consider a document as a unit of sentiment classification *i.e.* our goal is to predict a document on the whole as positive or negative. This is different from Pang and Lee (2004) which suggests that sentiment is associated only with subjective content. A document in its entirety is represented using sense-based features in our experiments. Carrillo de Albornoz et al. (2010) suggests expansion using WordNet relations which we also follow. This is a benefit that can be achieved only in a sense-based space.

3 Features based on WordNet Senses

In their original form, documents are said to be in lexical space since they consist of words. When the words are replaced by their corresponding senses, the resultant document is said to be in semantic space.

WordNet 2.1 (Fellbaum, 1998) has been used as the sense repository. Each word/lexeme is mapped to an appropriate synset in WordNet based on its sense and represented using the corresponding synset id of WordNet. Thus, the word *love* is disambiguated and replaced by the identifier *21758160* which consists of a POS category identifier *2* followed by synset offset identifier *1758160*. This paper refers to synset offset as synset identifiers or simply, senses.

This section first gives the motivation for using word senses and then, describes the approaches that we use for our experiments.

3.1 Motivation

Consider the following sentences as the first scenario.

1. *"Her face **fell** when she heard that she had been fired."*

2. *"The fruit **fell** from the tree."*

The word '*fell*' occurs in different senses in the two sentences. In the first sentence, '*fell*' has the meaning of '*assume a disappointed or sad expression*, whereas in the second sentence, it has the meaning of '*descend in free fall under the influence of gravity*'. A user will infer the negative polarity of the first sentence from the negative sense of '*fell*' in it while the user will state that the second sentence does not carry any sentiment. This implies that there is at least one sense of the word '*fell*' that carries sentiment and at least one that does not.

In the second scenario, consider the following examples.

1. *"The snake bite proved to be **deadly** for the young boy."*

2. *"Shane Warne is a **deadly** spinner."*

The word *deadly* has senses which carry opposite polarity in the two sentences and these senses assign the polarity to the corresponding sentence. The first sentence is negative while the second sentence is positive.

Finally in the third scenario, consider the following pair of sentences.

1. *"He speaks a **vulgar** language."*

2. *"Now that's real **crude** behavior!"*

The words *vulgar* and *crude* occur as synonyms in the synset that corresponds to the sense '*conspicuously and tastelessly indecent*'. The synonymous nature of words can be identified only if they are looked at as senses and not just words.

As one may observe, the first scenario shows that a word may have some sentiment-bearing and some non-sentiment-bearing senses. In the second scenario, we show that there may be different senses of a word that bear sentiments of opposite polarity. Finally, in the third scenario, we show how a sense can be manifested using different words, *i.e.*, words in a synset. The three scenarios motivate the use of semantic space for sentiment prediction.

3.2 Sense versus Lexeme-based Feature Representation

We annotate the words in the corpus with their senses using two sense disambiguation approaches.

As the first approach, **manual sense annotation** of documents is carried out by two annotators on two subsets of the corpus, the details of which are given in Section 5.1. This is done to determine the ideal case scenario- the skyline performance.

As the second approach, a state-of-art algorithm for domain-specific WSD proposed by Khapra et al. (2010) is used to obtain an automatically sense-tagged corpus. This algorithm called **iterative WSD or IWSD** iteratively disambiguates words by ranking the candidate senses based on a scoring function.

The two types of sense-annotated corpus lead us to four feature representations for a document:

1. Word senses that have been manually annotated (*M*)

2. Word senses that have been annotated by an automatic WSD (*I*)

3. *Manually* annotated word senses and words (both separately as features) (*Words + Sense(M)*)

4. *Automatically* annotated word senses and words (both separately as features) (*Words + Sense(I)*)

Our first set of experiments compares the four feature representations to find the feature representation with which sentiment classification gives the best performance. W+S(M) and W+S(I) are used to overcome non-coverage of WordNet for some noun synsets. In addition to this, we also present a part-of-speech-wise analysis of benefit to SA as well as effect of varying the training samples on sentiment classification accuracy.

3.3 Partial disambiguation as opposed to no disambiguation

The state-of-the-art automatic WSD engine that we use performs (approximately) with 70% accuracy on tourism domain (Khapra et al., 2010). This means that the performance of SA depends on the performance of WSD which is not very high in case of the engine we use.

A partially disambiguated document is a document which does not contain senses of all words. Our hypothesis is that disambiguation of even few words in a document can give better results than no disambiguation. To verify this, we create different variants of the corpus by disambiguating words which have candidate senses within a threshold. For example, a partially disambiguated variant of the corpus with threshold 3 for candidate senses is created by disambiguating words which have *a maximum of three candidate senses*. These synsets are then used as features for classification along with lexeme based features. We conduct multiple experiments using this approach by varying the number of candidate senses.

4 Advantage of senses: Similarity Metrics and Unknown Synsets

4.1 Synset Replacement Algorithm

Using WordNet senses provides an opportunity to use similarity-based metrics for WordNet to reduce the effect of unknown features. If a synset encountered in a test document is not found in the training corpus, it is replaced by one of the synsets present in the training corpus. The substitute synset is determined on the basis of its similarity with the synset in the test document. The synset that is replaced is referred to as an *unseen synset* as it is not known to the trained model.

For example, consider excerpts of two reviews, the first of which occurs in the training corpus while the second occurs in the test corpus.

1. " *In the night, it is a* ***lovely*** *city and...* "

2. " *The city has many* ***beautiful*** *hot spots for honeymooners.* "

The synset of '*beautiful*' is not present in the training corpus. We evaluate a similarity metric for all synsets in the training corpus with respect to the sense of *beautiful* and find that the sense of *lovely* is closest to it. Hence, the sense of *beautiful* in the test document is replaced by the sense of *lovely* which is present in the training corpus.

The replacement algorithm is described in Algorithm 1. The algorithm follows from the fact that the similarity value for a synset with itself is maximum.

4.2 Similarity metrics used

We conduct different runs of the replacement algorithm using three similarity metrics, namely LIN's similarity metric, Lesk similarity metric and Leacock and Chodorow (LCH) similarity metric. These runs generate three variants of the corpus. We compare the benefit of each of these metrics by studying their sentiment classification performance. The metrics can be described as follows:

LIN: The metric by Lin (1998) uses the information content individually possessed by two concepts in addition to that shared by them. The information content shared by two concepts A and B is given by their most specific subsumer (lowest superordinate(*lso*). Thus, this metric defines the similarity between two concepts as

$$sim_{LIN}(A,B) = \frac{2 \times \log Pr(lso(A,B))}{\log Pr(A) + \log Pr(B)} \quad (1)$$

```
Input: Training Corpus, Test Corpus,
Similarity Metric
Output: New Test Corpus
T:= Training Corpus;
X:= Test Corpus;
S:= Similarity metric;
train_concept_list = get_list_concept(T) ;
test_concept_list = get_list_concept(X);
for each concept C in test_concept_list do
    temp_max_similarity = 0 ;
    temp_concept = C ;
    for each concept D in train_concept_list do
        similarity_value = get_similarity_value(C,D,S);
        if (similarity_value > temp_max_similarity) then
            temp_max_similarity= similarity_value;
            temp_concept = D ;
        end
    end
    C = temp_concept ;
    replace_synset_corpus(C,X);
end
Return X ;
```

Algorithm 1: Synset replacement using similarity metric

Lesk: Each concept in WordNet is defined through gloss. To compute the Lesk similarity (Banerjee and Pedersen, 2002) between A and B, a scoring function based on the overlap of words in their individual glosses is used.

Leacock and Chodorow (LCH): To measure similarity between two concepts A and B, Leacock and Chodorow (1998) compute the shortest path through hypernymy relation between them under the constraint that there exists such a path. The final value is computed by scaling the path length by the overall taxonomy depth (D).

$$sim_{LCH}(A,B) = -\log\left(\frac{len(A,B)}{2D}\right) \quad (2)$$

5 Experimentation

We describe the variants of the corpus generated and the experiments in this section.

5.1 Data Preparation

We create different variants of the dataset by Ye et al. (2009). This dataset contains 600 positive and 591 negative reviews about seven travel destinations. Each review contains approximately 4-5 sentences with an average number of words per review being 80-85.

To create the manually annotated corpus, two human annotators annotate words in the corpus with senses for two disjoint subsets of the original corpus by Ye et al. (2009). The inter-annotation agreement for a subset of the corpus showed 91% sense overlap. The manually annotated corpus consists of 34508 words with 6004 synsets.

POS	#Words	P(%)	R(%)	F-Score(%)
Noun	12693	75.54	75.12	75.33
Adverb	4114	71.16	70.90	71.03
Adjective	6194	67.26	66.31	66.78
Verb	11507	68.28	67.97	68.12
Overall	34508	71.12	70.65	70.88

Table 1: Annotation Statistics for IWSD; P- Precision,R-Recall

The second variant of the corpus contains word senses obtained from automatic disambiguation using IWSD. The evaluation statistics of the IWSD is shown in Table 1. Table 1 shows that the F-score for noun synsets is high while that for adjective synsets is the lowest among all. The low recall for adjective POS based synsets can be detrimental to classification since adjectives are known to express direct sentiment (Pang et al., 2002). Hence, in the context of sentiment classification, disambiguation of adjective synsets is more critical as compared to disambiguation of noun synsets.

5.2 Experimental setup

The experiments are performed using C-SVM (linear kernel with default parameters[1]) available as a part of LibSVM[2] package. We choose to use SVM since it performs the best for sentiment classification (Pang et al., 2002). All results reported are average of five-fold cross-validation accuracies.

To conduct experiments on words as features, we first perform stop-word removal. The words are not stemmed since stemming is known to be detrimental to sentiment classification (Leopold and Kindermann, 2002). To conduct the experiments based on

[1] C=0.0,ϵ=0.0010

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

Feature Representations	Accuracy(%)	PF	NF	PP	NP	PR	NR
Words (Baseline)	84.90	85.07	84.76	84.95	84.92	85.19	84.60
Sense (M)	89.10	88.22	89.11	91.50	87.07	85.18	91.24
Words + Sense (M)	90.20	89.81	90.43	92.02	88.55	87.71	92.39
Sense (I)	85.48	85.31	85.65	87.17	83.93	83.53	87.46
Words + Sense (I)	86.08	86.28	85.92	85.87	86.38	86.69	85.46

Table 2: Classification Results; PF-Positive F-score(%), NF-Negative F-score (%), PP-Positive Precision (%), NP-Negative Precision (%), PR-Positive Recall (%), NR-Negative Recall (%)

the synset representation, words in the corpus are annotated with synset identifiers along with POS category identifiers. For automatic sense disambiguation, we used the trained IWSD engine from Khapra et al. (2010). These synset identifiers along with POS category identifiers are then used as features. For replacement using semantic similarity measures, we used WordNet::Similarity 2.05 package by Pedersen et al. (2004).

To evaluate the result, we use accuracy, F-score, recall and precision as the metrics. Classification accuracy defines the ratio of the number of true instances to the total number of instances. Recall is calculated as a ratio of the true instances found to the total number of false positives and true positives. Precision is defined as the number of true instances divided by number of true positives and false negatives. Positive Precision (PP) and Positive Recall (PR) are precision and recall for positive documents while Negative Precision (NP) and Negative Recall (NR) are precision and recall for negative documents. F-score is the weighted precision-recall score.

6 Results and Discussions

6.1 *Comparison of various feature representations*

Table 2 shows results of classification for different feature representations. The baseline for our results is the unigram bag-of-words model (Baseline).

An improvement of 4.2% is observed in the accuracy of sentiment prediction when manually annotated sense-based features (M) are used in place of word-based features (Words). The precision of both the classes using features based on semantic space is also better than one based on lexeme space. While reported results suggest that it is more difficult to detect negative sentiment than positive sentiment (Gindl and Liegl, 2008), our results show that negative recall increases by around 8% in case of sense-based representation of documents.

The combined model of words and manually annotated senses (Words + Senses (M)) gives the best performance with an accuracy of 90.2%. This leads to an improvement of 5.3% over the baseline accuracy [3].

One of the reasons for improved performance is the feature abstraction achieved due to the synset-based features. The dimension of feature vector is reduced by a factor of 82% when the document is represented in synset space. The reduction in dimensionality may also lead to reduction in noise (Cunningham, 2008).

A comparison of accuracy of different sense representations in Table 2 shows that manual disambiguation performs better than using automatic algorithms like IWSD. Although overall classification accuracy improvement of IWSD over baseline is marginal, negative recall also improves. This benefit is despite the fact that evaluation of IWSD engine over manually annotated corpus gave an overall F-score of 71% (refer Table 1). For a WSD engine with a better accuracy, the performance of sense-based SA can be boosted further.

Thus, in terms of feature representation of documents, sense-based features provide a better overall performance as compared to word-based features.

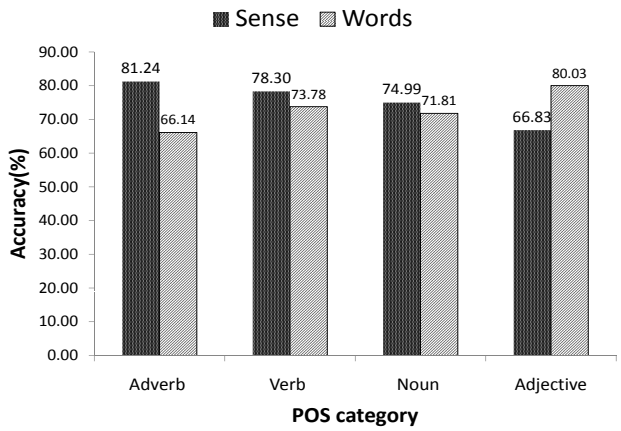


Figure 1: POS-wise statistics of manually annotated semantic space

6.2 *POS-wise analysis*

For each POS, we compare the performance of two models:

- Model trained on words of only that POS
- Model trained on word senses of only that POS

Figure 1 shows the parts-of-speech-wise classification accuracy of sentiment classification for senses (manual) and words. In the lexeme space, adjectives directly impact the classification performance. But it can be seen that disambiguation of adverb and verb synsets impact the performance of SA higher than disambiguation of nouns and adjectives.

While it is believed that adjectives carry direct sentiments, our results suggest that using adjectives alone as features may not improve the accuracy. The results prove that sentiment may be subtle at times and not expressed directly through adjectives.

As manual sense annotation is an effort and cost intensive process, the parts-of-speech-wise results suggest improvements expected from an automatic WSD engine so that it can aid sentiment classification. Table 1 suggests that the WSD engine works better for noun synsets compared to adjective and adverb synsets. While this is expected in a typical WSD setup, it is the adverbs and verbs that are more important for detecting sentiment in semantics space than nouns. The future WSD systems will have to show an improvement in their accuracy with respect to adverb and verb synsets.

	Sense		Words	
POS Category	**PF**	**NF**	**PF**	**NF**
Adverb	79.65	80.45	70.25	73.68
Verb	75.50	79.28	62.23	63.12
Noun	73.39	75.40	69.77	72.55
Adjective	63.11	65.03	78.29	79.20

Table 3: POS-wise F-score for sense (M) and Words;PF-Positive F-score(%), NF- Negative F-score (%)

Table 3 shows the positive and negative F-score statistics with respect to different POS. Detection of negative reviews using lexeme space is difficult. POS-wise statistics also suggest the same. It should be noted that adverb and verb synsets play an important role in negative class detection. Thus, an automatic WSD engine should give importance to the correct disambiguation of these POS categories.

6.3 *Effect of size of training corpus*

#Training Documents	**W**	**M**	**I**	**W+S(M)**	**W+S(I)**
100	76.5	87	79.5	82.5	79.5
200	81.5	88.5	82	90	84
300	79.5	92	81	89.5	82
400	82	90.5	81	94	85.5
500	83.5	91	85	96	82.5

Table 4: Accuracy (%) with respect to number of training documents; W: Words, M: Manual Annotation, I: IWSD-based sense annotation, W+S(M): Word+Senses (Manual annotation), W+S(I): Word+Senses(IWSD-based sense annotation)

From table 2, the benefit of sense disambiguation to sentiment prediction is evident. In addition, Table 4 shows variation of classification accuracy with respect to different number of training samples based on different approaches of annotation explained in previous sections. The results are based on a blind set of 90 test samples from both the polarity labels [4].

[3]The improvement in results of semantic space is found to be statistically significant over the baseline at 95% confidence level when tested using a paired t-test.

[4]No cross validation is performed for this experiment

Compared to lexeme-based features, manually annotated sense based features give better performance with lower number of training samples. IWSD is also better than lexeme-based features. A SA system trained on 100 training samples using manually annotated senses gives an accuracy of 87%. Word-based features never achieve this accuracy. An IWSD-based system requires lesser samples when compared to lexeme space for an equivalent accuracy. Note that model based on words + senses(M) features achieve an accuracy of 96% on this test set.

This implies that the synset space, in addition to benefit to sentiment prediction in general, requires *lesser number of training samples* in order to achieve the accuracy that lexeme space can achieve with a larger number of samples.

6.4 *Effect of Partial disambiguation*

Figure 2 shows the accuracy, positive F-score and negative F-score with respect to different thresholds of candidate senses for partially disambiguated documents as described in Section 3.3. We compare the performance of these documents with word-based features (B) and sense-based features based on manually (M) or automatically obtained senses (I). Note that Sense (I) and Sense (M) correspond to completely disambiguated documents.

In case of partial disambiguation using manual annotation, disambiguating words with less than three candidate senses performs better than others. For partial disambiguation that relies on an automatic WSD engine, a comparable performance to full disambiguation can be obtained by disambiguating words which have a maximum of four candidate senses.

As expected, completely disambiguated documents provide the best F-score and accuracy figures[5]. However, a performance comparable to complete disambiguation can be attained by disambiguating selective words.

Our results show that even if highly ambiguous (in terms of senses) words are not disambiguated by a WSD engine, the performance of sentiment classification improves.

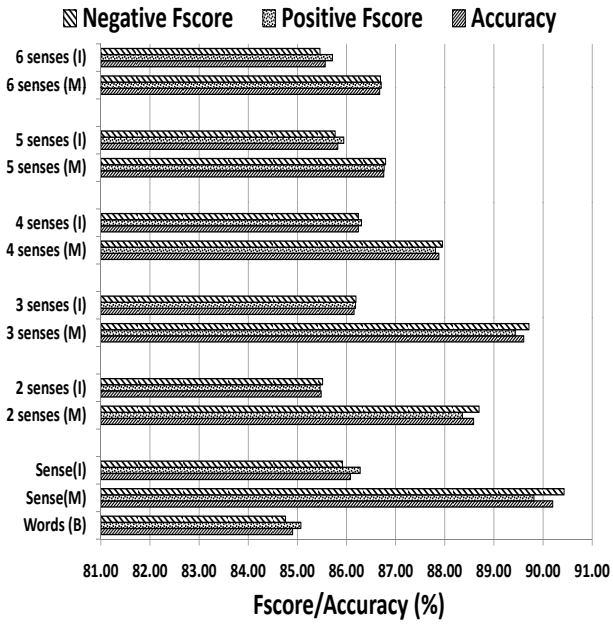


Figure 2: Partial disambiguation statistics: Accuracy,Positive F-score, Negative F-score variation with respect to sense disambiguation difficult level is shown. Words(B): baseline system

6.5 *Synset replacement using similarity metrics*

Table 5 shows the results of synset replacement experiments performed using similarity metrics defined in section 4. The similarity metric value NA shown in the table indicates that synset replacement is not performed for the specific run of experiment. For this set of experiments, we use the combination of sense and words as features (indicated by *Senses+Words (M)*).

Synset replacement using a similarity metric shows an improvement over using words alone. However, the improvement in classification accuracy is marginal compared to sense-based representation without synset replacement (Similarity Metric=NA).

Replacement using LIN and LCH metrics gives marginally better results compared to the vanilla setting in a manually annotated corpus. The same phenomenon is seen in the case of IWSD based approach[6]. The limited improvement can be due to the fact that since LCH and LIN consider only IS-A

[5] All results are statistically significant with respect to baseline

[6] Results based on LCH and LIN similarity metric for automatic sense disambiguation is not statistically significant with α=0.05

Feature Representation	Similarity Metric	Accuracy	PF	NF	PP	NP	PR	NR
Words (Baseline)	**NA**	84.90	85.07	84.76	84.95	84.92	85.19	84.60
Words + Sense(M)	**NA**	90.20	89.81	90.43	92.02	88.55	87.71	92.39
Words + Sense(I)	**NA**	86.08	86.28	85.92	85.87	86.38	86.69	85.46
Words + Sense (M)	**LCH**	90.60	90.20	90.85	92.85	88.61	87.70	93.21
Words + Sense(M)	**LIN**	90.70	90.26	90.97	93.17	88.50	87.53	93.57
Words + Sense (M)	**Lesk**	91.12	90.70	91.38	93.55	88.97	88.03	93.92
Words + Sense (I)	**LCH**	85.66	85.85	85.52	85.67	85.76	86.02	85.28
Words + Sense(I)	**LIN**	86.16	86.37	86.00	86.06	86.40	86.69	85.61
Words + Sense (I)	**Lesk**	86.25	86.41	86.10	86.31	86.26	86.52	85.95

Table 5: Similarity Metric Analysis using different similarity metrics with synsets and a combinations of synset and words;PF-Positive F-score(%), NF-Negative F-score (%), PP-Positive Precision (%), NP-Negative Precision (%), PR-Positive Recall (%), NR-Negative Recall (%)

Top information content features (in %)	IWSD synset #	Manual synsets #	Match synset #	Match Synsets (%)	Unmatched Synset(%)
10	601	722	288	39.89	60.11
20	1199	1443	650	45.05	54.95
30	1795	2165	1005	46.42	53.58
40	2396	2889	1375	47.59	52.41
50	2997	3613	1730	47.88	52.12

Table 6: Comparison of top information gain-based features of manually annotated corpora and automatically annotated corpora

relationship in WordNet, the replacement happens only for verbs and nouns. This excludes adverb synsets which we have shown to be the best features for a sense-based SA system.

Among all similarity metrics, the best classification accuracy is achieved using Lesk. The system performs with an overall classification accuracy of 91.12%, which is a substantial improvement of 6.2% over baseline. Again, it is only 1% over the vanilla setting that uses combination of synset and words. However, the similarity metric is not sophisticated as LIN or LCH.

Thus, we observe a marginal improvement by using similarity-based metrics for WordNet. A good metric which covers all POS categories can provide substantial improvement in the classification accuracy.

7 Error Analysis

For sentiment classification based on semantic space, we classify the errors into four categories. The examples quoted are from manual evaluation of the results.

1. *Effect of low disambiguation accuracy of IWSD engine:* SA using automatic sense annotation depends on the annotation system used. To assess the impact of IWSD system on sentiment classification, we compare the feature set based on manually annotated senses with the feature set based on automatically annotated senses. We compare the most informative features of the two classifiers. Table 6 shows the number of top informative features (synset) selected as the percentage of total synset features present when the semantic representation of documentation is used. The matched synset column represents the number of IWSD synsets that match

with manually annotated synsets.

The number of top performing features is more in case of manually annotated synsets. This can be attributed to the total number of synsets tagged in the two variant of the corpus. The reduction in the performance of SA for automatically annotated senses is because of the number of unmatched synsets.

Thus, although the accuracy of IWSD is currently 70%, the table indicates that IWSD can match the performance of manually annotated senses for SA if IWSD is able to tag correctly those top information content synsets. This aspect needs to be investigated further.

2. *Negation Handling:* For the purpose of this work, we concentrate on words as units for sentiment determination. Syntax and its contribution in understanding sentiment is neglected and hence, positive documents which contain negations are wrongly classified as negative. Negation may be direct as in the excerpt '*....what is there not to like about Vegas.*' or may be double as in the excerpt'*...that aren't insecure'*.

3. *Interjections and WordNet coverage:* Recent informal words are not covered in WordNet and hence, do not get disambiguated. The same is the case for interjections like *'wow'*,*'duh'* which sometimes carry direct sentiment. Lexical resources which include them can be used to incorporate information about these lexical units.

4. *Document Specificity:* The assumption underlying our analysis is that a document contains description of only one topic. However, reviews are generic in nature and tend to express contrasting sentiment about sub-topics . For example, a travel review about Paris can talk about restaurants in Paris, traffic in Paris, public behaviour, etc. with opposing sentiments. Assigning an overall sentiment to a document is subjective in such cases.

8 Conclusion & Future Work

This work presents an empirical benefit of WSD to sentiment analysis. The study shows that supervised sentiment classifier modeled on wordNet senses perform better than word-based features. We show how the performance impact differs for different automatic and manual techniques, parts-of-speech, different training sample size and different levels of disambiguation. In addition, we also show the benefit of using WordNet based similarity metrics for replacing unknown features in the test set. Our results support the fact that not only does sense space improve the performance of a sentiment classification system, but also opens opportunities for improvement using better similarity metrics.

Incorporation of syntactical information along with semantics can be an interesting area of work. More sophisticated features which include the two need to be explored. Another line of work is in the context of cross-lingual sentiment analysis. Current solutions are based on machine translation which is very resource-intensive. Using a bi-lingual dictionary which maps WordNet across languages, such a machine translation sub-system can be avoided.

Acknowledgment

We thank Jaya Saraswati and Rajita Shukla from CFILT Lab, IIT Bombay for annotating the dataset used for this work. We also thank Mitesh Khapra and Salil Joshi, IIT Bombay for providing us with the IWSD engine for the required experiments.

References

Cem Akkaya, Janyce Wiebe, and Rada Mihalcea. 2009. Subjectivity word sense disambiguation. In *Proc. of EMNLP '09*, pages 190–199, Singapore.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Proc. of CICLing'02*, pages 136–145, London, UK.

Jorge Carrillo de Albornoz, Laura Plaza, and Pablo Gervs. 2010. Improving emotional intensity classification using word sense disambiguation. *Special issue: Natural Language Processing and its Applications. Journal on Research in Computing Science*, 46:131–142.

Pdraig Cunningham. 2008. Dimension reduction. In *Machine Learning Techniques for Multimedia*, Cognitive Technologies, pages 91–112.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Stefan Gindl and Johannes Liegl, 2008. *Evaluation of different sentiment detection methods for polarity classification on web-based reviews*, pages 35–43.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125.

Mitesh Khapra, Sapan Shah, Piyush Kedia, and Pushpak Bhattacharyya. 2010. Domain-specific word sense disambiguation combining corpus basedand wordnet based parameters. In *Proc. of GWC'10*, Mumbai, India.

Claudia Leacock and Martin Chodorow. 1998. Combining local context with wordnet similarity for word sense identification. In *WordNet: A Lexical Reference System and its Application*.

Edda Leopold and Jörg Kindermann. 2002. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46:423–444.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *In Proc. of the 15th International Conference on Machine Learning*, pages 296–304.

Tamara Martn-Wanton, Alexandra Balahur-Dobrescu, Andres Montoyo-Guijarro, and Aurora Pons-Porrata. 2010. Word sense disambiguation in opinion mining: Pros and cons. In *Proc. of CICLing'10*, Madrid,Spain.

Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. 2005. Sentiment classification using word sub-sequences and dependency sub-trees. In *Proc. of PAKDD'05,*, Lecture Notes in Computer Science, pages 301–311.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of ACL'04*, pages 271–278, Barcelona, Spain.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2:1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. volume 10, pages 79–86.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL'04*, pages 38–41.

Vassiliki Rentoumi, George Giannakopoulos, Vangelis Karkaletsis, and George A. Vouros. 2009. Sentiment analysis of figurative language using a word sense disambiguation approach. In *Proc. of the International Conference RANLP'09*, pages 370–375, Borovets, Bulgaria.

Peter Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proc. of ACL'02*, pages 417–424, Philadelphia, US.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. 2005. Using appraisal groups for sentiment analysis. In *Proc. of CIKM '05*, pages 625–631, New York, NY, USA.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Proc. of COLING-ACL'06*, pages 1065–1072.

Qiang Ye, Ziqiong Zhang, and Rob Law. 2009. Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert Systems with Applications*, 36(3):6527 – 6535.

Learning General Connotation of Words using Graph-based Algorithms

Song Feng **Ritwik Bose** **Yejin Choi**
Department of Computer Science
Stony Brook University
NY 11794, USA
songfeng, rbose, ychoi@cs.stonybrook.edu

Abstract

In this paper, we introduce a *connotation lexicon*, a new type of lexicon that lists words with connotative polarity, i.e., words with positive connotation (e.g., award, promotion) and words with negative connotation (e.g., cancer, war). Connotation lexicons differ from much studied sentiment lexicons: the latter concerns words that *express* sentiment, while the former concerns words that *evoke* or *associate with* a specific polarity of sentiment. Understanding the connotation of words would seem to require common sense and world knowledge. However, we demonstrate that much of the connotative polarity of words can be inferred from natural language text in a nearly unsupervised manner. The key linguistic insight behind our approach is *selectional preference* of *connotative predicates*. We present graph-based algorithms using PageRank and HITS that collectively learn connotation lexicon together with connotative predicates. Our empirical study demonstrates that the resulting connotation lexicon is of great value for sentiment analysis complementing existing sentiment lexicons.

1 Introduction

In this paper, we introduce a ***connotation lexicon***, a new type of lexicon that lists words with connotative polarity, i.e., words with positive connotation (e.g., award, promotion) and words with negative connotation (e.g., cancer, war). Connotation lexicons differ from sentiment lexicons that are studied in much of previous research (e.g., Esuli and Sebastiani (2006), Wilson et al. (2005a)): the latter concerns words that *express* sentiment either explicitly or implicitly, while the former concerns words that *evoke* or even simply *associate with* a specific polarity of sentiment. To our knowledge, there has been no previous research that investigates polarized connotation lexicons.

Understanding the connotation of words would seem to require common sense and world knowledge at first glance, which in turn might seem to require human encoding of knowledge base. However, we demonstrate that much of the connotative polarity of words can be inferred from natural language text in a nearly unsupervised manner.

The key linguistic insight behind our approach is *selectional preference* of *connotative predicates*. We define a ***connotative predicate*** as a predicate that has selectional preference on the connotative polarity of some of its semantic arguments. For instance, in the case of the connotative predicate *"prevent"*, there is strong selectional preference on negative connotation with respect to the thematic role (semantic role) "THEME". That is, statistically speaking, people tend to associate negative connotation with the THEME of *"prevent"*, e.g., *"prevent cancer"* or *"prevent war"*, rather than positive connotation, e.g., *"prevent promotion"*. In other words, even though it is perfectly valid to use words with positive connotation in the THEME role of *"prevent"*, statistically more dominant connotative polarity is negative. Similarly, the THEME of *"congratulate"* or *"praise"* has strong selectional preference on positive connotation.

The theoretical concept supporting the selective

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1092–1103,
Edinburgh, Scotland, UK, July 27–31, 2011.

accomplish, achieve, advance, advocate, admire, applaud, appreciate, compliment, congratulate, develop, desire, enhance, enjoy, improve, praise, promote, respect, save, support, win

Table 1: Positively Connotative Predicates w.r.t. THEME

alleviate, accuse, avert, avoid, cause, complain, condemn, criticize, detect, eliminate, eradicate, mitigate, overcome, prevent, prohibit, protest, refrain, suffer, tolerate, withstand

Table 2: Negatively Connotative Predicates w.r.t. THEME

preference of connotative predicates is that of semantic prosody in corpus linguistics. Semantic prosody describes how some of the seemingly neutral words (e.g., "cause") can be perceived with positive or negative polarity because they tend to collocate with words with corresponding polarity (e.g., Sinclair (1991), Louw et al. (1993), Stubbs (1995), Stefanowitsch and Gries (2003)). In this work, we demonstrate that statistical approaches that exploit this very concept of semantic prosody can successfully infer connotative polarity of words.

Having described the key linguistic insight, we now illustrate our graph-based algorithms. Figure 1 depicts the mutually reinforcing relation between connotative predicates (nodes on the left-hand side) and words with connotative polarity (node on the right-hand side). The thickness of edges represents the strength of the association between predicates and arguments. For brevity, we only consider connotation of words that appear in the THEME thematic role.

We expect that words that appear often in the THEME role of various positively (or negatively) connotative predicates are likely to be words with positive (or negative) connotation. Likewise, predicates whose THEME contains words with mostly positive (or negative) connotation are likely to be positively (or negatively) connotative predicates. In short, we can induce the connotative polarity of words using connotative predicates, and inversely, we can learn new connotative predicates based on words with connotative polarity.

We hypothesize that this mutually reinforcing relation between connotative predicates and their arguments can be captured via graph centrality in graph-based algorithms. Given a small set of seed words for connotative predicates, our algorithms collectively learn connotation lexicon together with connotative predicates in a nearly unsupervised manner. A number of different graph representations are explored using both PageRank (Page et al., 1999) and HITS (Kleinberg, 1999) algorithms. Empirical study demonstrates that our graph based algorithms are highly effective in learning both connotation lexicon and connotative predicates.

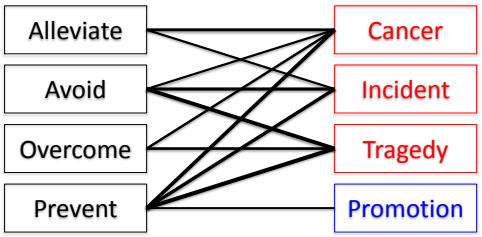


Figure 1: Bipartite graph of connotative predicates and arguments. Edge weights are proportionate to the association strength.

Finally, we quantify the practical value of our connotation lexicon in concrete sentiment analysis applications, and demonstrate that the connotation lexicon is of great value for sentiment classification tasks complementing conventional sentiment lexicons.

2 Connotation Lexicon & Connotative Predicate

In this section, we define *connotation lexicon* and *connotative predicates* more formally, and contrast them against words in conventional sentiment lexicons.

2.1 Connotation Lexicon

This lexicon lists words with positive and negative connotation, as defined below.

- **Words with positive connotation:** In this work, we define words with positive connotation as those that describe physical objects or abstract concepts that people generally value, cherish or care about. For instance, we regard words such as "freedom", "life", or "health" as

words with positive connotation. Some of these words may express subjectivity either explicitly or implicitly, e.g., "joy" or "satisfaction". However, a substantial number of words with positive connotation are purely objective, such as "life", "health", "tenure", or "scientific".

- **Words with negative connotation:** We define words with negative connotation as those that describe physical objects or abstract concepts that people generally disvalue or avoid. Similarly as before, some of these words may express subjectivity (e.g., "disappointment", "humiliation"), while many other are purely objective (e.g., "bedbug", "arthritis, "funeral").

Note that this explicit and intentional inclusion of objective terms makes connotation lexicons differ from sentiment lexicons: most conventional sentiment lexicons have focused on subjective words by definition (e.g., Wilson et al. (2005b)), as many researchers use the term *sentiment* and *subjectivity* interchangeably (e.g., Wiebe et al. (2005)).

2.2 Connotative Predicate

In this work, connotative predicates are those that exhibit selectional preference on the connotative polarity of some of their arguments. We emphasize that the polarity of connotative predicates does *not* coincide with the polarity of sentiment in conventional sentiment lexicons, as will be elaborated below.

- **Positively connotative predicate:** In this work, we define positively connotative predicates as those that expect positive connotation in some arguments. For example, "congratulate" or "save" are positively connotative predicates that expect words with positive connotation in the THEME argument: people typically congratulate something positive, and save something people care about. More examples are shown in Table 1.
- **Negatively connotative predicate:** In this work, we define negatively connotative predicates as those that expect negative connotation in some arguments. For instance, predicates such as "prevent" or "suffer" tend to project negative connotation in the THEME argument. More examples are shown in Table 2.

Note that positively connotative predicates are not necessarily positive sentiment words. For instance "save" is not a positive sentiment word in the lexicon published by Wilson et al. (2005b). Inversely, (strongly) positive sentiment words are not necessarily (strongly) positively connotative predicates, e.g., "illuminate", "agree". Likewise, negatively connotative predicates are not necessarily negative sentiment words. For instance, predicates such as "prevent", "detect", or "cause" are not negative sentiment words, but they tend to correlate with negative connotation in the THEME argument. Inversely, (strongly) negative sentiment words are not necessarily (strongly) negatively connotative predicates, e.g., "abandon" ("abandoned [something valuable]").

3 Graph Representation

In this section, we explore the graphical representation of our task. Figure 1 depicts the key intuition as a bipartite graph, where the nodes on the left-hand side correspond to connotative predicates, and the nodes on the right-hand side correspond to words in the THEME argument. There is an edge between a predicate p and an argument a, if the argument a appears in the THEME role of the predicate p. For brevity, we explore only verbs as the predicates, and words in the THEME role of the predicates as arguments. Our work can be readily extended to exploit other predicate-argument relations however.

Note that there are many sources of noise in the construction of graph. For instance, some of the predicates might be negated, changing the semantic dynamics between the predicate and the argument. In addition, there might be many unusual combinations of predicates and arguments, either due to data processing errors or due to idiosyncratic use of language. Some of such combinations can be valid ones (e.g., "prevent promotion"), challenging the learning algorithm with confusing evidence.

We hypothesize that by focusing on the important part of the graph via centrality analysis , it is possible to infer connotative polarity of words despite various noise introduced in the graph structure. This implies that it is important to construct the graph structure so as to capture important linguistic relations between predicates and arguments. With this goal in mind,

we next explore the directionality of the edges and different strategies to assign weights to them.

3.1 Undirected (Symmetric) Graph

First we explore undirected edges. In this case, we assign weight for each undirected edge between a predicate p and an argument a. Intuitively, the weight should correspond to the strength of relatedness or association between the predicate p and the argument a. We use Pointwise Mutual Information (PMI), as it has been used by many previous research to quantify the association between two words (e.g., Turney (2001), Church and Hanks (1990)). The PMI score between p and a is defined as follows:

$$w(p-a) := PMI(p,a) = log\frac{P(p,a)}{P(p)P(a)}$$

The log of the ratio is positive when the pair of words tends to co-occur and negative when the presence of one word correlates with the absence of the other word.

3.2 Directed (Asymmetric) Graph

Next we explore directed edges. That is, for each connected pair of a predicate p and an argument a, there are two edges in opposite directions: $e(p \rightarrow a)$ and $e(a \rightarrow p)$. In this case, we explore the use of asymmetric weights using conditional probability. In particular, we define weights as follows:

$$w(p \rightarrow a) := P(a|p) = \frac{P(p,a)}{P(p)}$$

$$w(a \rightarrow p) := P(p|a) = \frac{P(p,a)}{P(a)}$$

Having defined the graph structure, next we explore algorithms that analyze graph centrality via random walks. In particular, we investigate the use of HITS algorithm (Section 4), and PageRank (Section 5).

4 Lexicon Induction using HITS

The graph representation described thus far (Section 3) captures general semantic relations between predicates and arguments, rather than those specific to connotative predicates and arguments. Therefore in this section, we explore techniques to augment the graph representation so as to bias the centrality of the graph toward connotative predicates and arguments.

In order to establish a learning bias, we start with a small set of seed words for *just* connotative predicates. We use 20 words for each polarity, as listed in Table 1 and Table 2. These seed words act as prior knowledge in our learning. We explore two different techniques to incorporate prior knowledge into random walk, as will be elaborated in Section 4.2 & 4.3, followed by brief description of HITS in Section 4.1.

4.1 Hyperlink-Induced Topic Search (HITS)

HITS (Hyperlink-Induced Topic Search) algorithm (Kleinberg, 1999), also known as Hubs and authorities, is a link analysis algorithm that is particularly suitable to model mutual reinforcement between two different types of nodes: hubs and authorities. The definitions of hubs and authorities are given recursively. A (good) hub is a node that points to many (good) authorities, and a (good) authority is a node pointed by many (good) hubs.

Notice that the mutually reinforcing relationship is precisely what we intend to model between connotative predicates and arguments. Let $G = (P, A, E)$ be the bipartite graph, where P is the set of nodes corresponding to connotative predicates, A is the set of nodes corresponding to arguments, and E is the set of edges among nodes. $(P_i, A_j) \in E$ if and only if the predicate P_i and the argument A_i occur together as a predicate – argument pair in the corpus. The co-occurrence matrix derived from our corpus is denoted as L, where

$$L_{ij} = \begin{cases} w(i,j) & if(P_i, A_j) \in E \\ 0 & otherwise \end{cases}$$

The value of $w(i,j)$ is set to $w(i-j)$ as defined in Section 3.1 for undirected graphs, and $w(i \rightarrow j)$ defined in Section 3.2 for directed graphs.

Let $a(A_i)$ and $h(A_i)$ be the authority and hub score respectively, for a given node $A_i \in A$. Then we compute the authority and hub score recursively as follows:

$$a(A_i) = \sum_{P_i, A_j \in E} w(i,j)h(A_j) + \sum_{P_j, A_i \in E} h(P_j)w(j,i)$$

$$h(A_i) = \sum_{P_i, A_j \in E} w(i,j)a(A_j) + \sum_{P_j, A_i \in E} a(P_j)w(j,i)$$

The scores $a(P_i)$ and $h(P_i)$ for $P_i \in P$ are defined similarly as above.

In what follows, we describe two different techniques to incorporate prior knowledge. Note that it is possible to apply each of the following techniques to both directed and undirected graph representations introduced in Section 3. Also note that for each technique, we construct two separate graphs G^+ and G^- corresponding to positive and negative polarity respectively. That is, G^+ learns positively connotative predicates and arguments, while G^- learns negatively connotative predicates and arguments.

4.2 Prior Knowledge via Truncated Graph

First we introduce a method based on graph truncation. In this method, when constructing the bipartite graph, we limit the set of predicates P to only those words in the seed set, instead of including all words that can be predicates. In a way, the truncated graph representation can be viewed as the query induced graph on which the original HITS algorithm was invented (Kleinberg, 1999).

The truncated graph is very effective in reducing the level of noise that can be introduced by predicates of the opposite polarity. It may seem like we cannot discover new connotative predicates in the truncated graph however, as the graph structure is limited only to the seed predicates. We address this issue by alternating truncation to different side of the graph, i.e., left (predicates) or right (arguments), as illustrated in Figure 1, through multiple rounds of HITS.

For instance, we start with the graph $G = (P^o, A, E(P^o))$ that is truncated only on the left-hand side, with the seed predicates P^o. Here, $E(P^o)$ denotes the reduced set of edges discarding those edges that connect to predicates not in P^o. Then, we apply HITS algorithm until convergence to discover new words with connotation, and this completes the first round of HITS.

Next we begin the second round. Let A^o be the new words with connotation that are found in the first round. We now set A^o as seed words for the second phase of HITS, where we construct a new graph $G = (P, A^o, E(A^o))$ that is truncated only on the right-hand side, with full candidate words for predicates included on the left-hand side. This alternation can be repeated multiple times to discover many new connotative predicates and arguments.

4.3 Prior Knowledge via Focussed Graph

In the truncated graph described above, one potential concern is that the discovery of new words with connotation is limited to those that happen to correlate well with the seed predicates. To mitigate this problem, we explore an alternative technique based on the full graph, which we will name as *focussed graph*.

In this method, instead of truncating the graph, we simply emphasize the important portion of the graph via edge weights. That is, we assign high weights to those edges that connect a seed predicate with an argument, while assigning low weights for those edges that connect to a predicate outside the seed set. This way, we allow predicates not in the seed set to participate in hubs and authority scores, but in a much suppressed way. This method can be interpreted as a smoothed version of the truncated graph described in Section 4.2.

More formally, if the node A_i is connected to the seed predicate P_j, the value of co-occurrence matrix L_{ij} is defined by prior knowledge(e.g. $PMI(A_i, P_j)$ or $P(A_i|P_j)$), otherwise a small constant ϵ is assigned to the edge.

$$L_{ij} = \begin{cases} w(i,j) & if P_j \in E^o \\ \epsilon & otherwise \end{cases}$$

Similarly to the truncated graph, we proceed with multiple rounds of HITS, focusing different part of the bipartite graph alternately.

5 Lexicon Induction using PageRank

In this section, we explore the use of another popular approach for link analysis: PageRank (Page et al., 1999). We first describe PageRank algorithm briefly in Section 5.1, then introduce two different techniques to incorporate prior knowledge in Section 5.2 and 5.3.

5.1 PageRank

Let $G = (V, E)$ be the graph, where $v_i \in V = P \cup A$ are nodes (words) for the disjunctive set of predicates (P) and arguments (A), and $e_{(i,j)} \in E$ are edges. Let $In(i)$ be the set of nodes with an edge leading to n_i and similarly, $Out(i)$ be the set

of nodes that n_i has an edge leading to. At a given iteration of the algorithm, we update the score of n_i as follows:

$$S(i) = \alpha \sum_{j \in In(i)} S(j) \times \frac{w(i,j)}{|Out(i)|} + (1 - \alpha) \quad (1)$$

where the value α is constant *damping factor*. The value of α is typically set to 0.85. The value of $w(i,j)$ is set to $w(i-j)$ as defined in Section 3.1 for undirected graphs, and $w(i \rightarrow j)$ as defined in Section 3.2 for directed graphs. As before, we will consider two different techniques to incorporate prior knowledge into the graph analysis as follows.

5.2 Prior Knowledge via Truncated Graph

Unlike HITS, which was originally invented for a query-induced graph, PageRank is typically applied to the full graph. However, we can still apply the truncation technique introduced in Section 4.2 to PageRank as well. To do so, when constructing the bipartite graph, we limit the set of predicates P to only those words in the seed set, instead of including all words that can be predicates. Graph truncation eliminates the noise that can be introduced by predicates of the opposite polarity. However, in order to learn new predicates, we need to perform multiple rounds of PageRank, truncating different side of the bipartite graph alternately. Refer to Section 4.2 for futher details.

5.3 Prior Knowledge via Teleportation

We next explore what is known as teleportation technique for topic sensitive PageRank (Haveliwala, 2002). For this, we use the following equation that is slightly augmented from Equation 1.

$$S(i) = \alpha \sum_{j \in In(i)} S(j) \times \frac{w(i,j)}{|Out(i)|} + (1 - \alpha)\, \epsilon_i \quad (2)$$

Here, the new term ϵ_i is a *smoothing factor* that prevents cliques in the graph from garnering reputation through feedback (Bianchini et al. (2005)). In order to emphasize important portion of the graph, i.e., subgraphs connected to the seed set, we assign non-zero ϵ scores to only those important nodes, i.e., the seed set. Intuitively, this will cause the random walk to restart from the seed set with $(1-\alpha)$ = 0.15 probability for each step.

6 The Use of Google Web 1T Data

In order to implement the network of connotative predicates and arguments, we need a substantially large amount of documents. The quality of the co-occurrence statistics is expected to be proportionate to the size of corpus, but collecting and processing such a large amount of data is not trivial. We therefore resort to the Google Web 1T data (Brants and Franz., 2006), which consists of Google n-gram counts (frequency of occurrence of each n-gram) for $1 \leq n \leq 5$. The use of Web 1T data will lessen the challenge with respect to data acquisition, while still allowing us to enjoy the co-occurrence statistics of web-scale data. Because Web 1T data is just n-gram statistics, rather than a collection of normal documents, it does not provide co-occurrence statistics of any random word pairs. However, it provides a nice approximation to the particular co-occurrence statistics we are interested in, which are, predicate – argument pairs. This is because the THEME argument of a verb predicate is typically on the right hand side of the predicate, and the argument is within the close range of the predicate.

We now describe how to derive co-occurrence statistics of each predicate – argument pair using the Web 1T data. For a given predicate p and an argument a, we add up the count (frequency) of all n-grams ($2 \leq n \leq 5$) that match the following pattern:

$$[p]\ [\star]^{n-2}\ [a]$$

where p must be the first word (head), a must be the last word (tail), and $[\star]^{n-2}$ matches any $n-2$ number of words between p and a. Note that this rule enforces the argument a to be on the right hand side of the predicate p. To reduce the level of noise, we do not allow the wildcard $[\star]$ to match any punctuation mark, as such n-grams are likely to cross sentence boundaries representing invalid predicate – argument relations. We consider a word as a predicate if it is tagged as a verb by a Part-of-Speech tagger (Toutanova and Manning, 2000). For argument $[a]$, we only consider content-words.

The use of web n-gram statistics necessarily invites certain kinds of noise. For instance, some of the $[p]\ [\star]^{n-2}\ [a]$ patterns might not correspond to a valid predicate – argument relation. However, we expect that our graph-based algorithms — HITS and

Lexicon	FREQ	HITS-sT	HITS-aT	HITS-sF	HITS-aF	Page-aT	Page-aF
Top 100	73.6	67.8	**77.7**	67.8	48.4	76.3	**77.0**
Top 1000	67.8	60.6	**68.8**	60.6	38.0	68.4	**68.5**
Top MAX	**65.8**	57.6	**66.5**	57.6	39.1	65.5	65.7

Table 3: Comparison Result with General Inquirer Lexicon(%)

Lexicon	FREQ	HITS-sT	HITS-aT	HITS-sF	HITS-aF	Page-aT	Page-aF
Top 100	83.0	79.3	**86.3**	79.3	55.8	86.3	**87.2**
Top 1000	80.3	67.3	**81.3**	67.3	46.5	**80.7**	80.3
Top MAX	71.5	62.7	**72.2**	62.7	45.4	71.1	**72.3**

Table 4: Comparison Result with OpinionFinder (%)

PageRank — will be able to discern valid relations from noise, by focusing on the important part of the graph. In other words, we expect that good predicates will be supported by good arguments, and vice versa, thereby resulting in a reliable set of predicates and arguments that are mutually supported by each other.

7 Experiments

As a baseline, we use a simple method dubbed *FREQ*, which uses co-occurrence frequency with respect to the seed predicates. Using the pattern $[p]\ [\star]^{n-2}\ [a]$ (see Section 6), we collect two sets of n-gram records: one set using the positive connotative predicates, and the other using the negative connotative predicates. With respect to each set, we calculate the following for each word a,

- Given [a], the number of unique $[p]$ as $f1$
- Given [a], the number of unique phrases $[\star]^{n-2}$ as $f2$
- The number of occurrences of $[a]$ as $f3$

We then obtain the score σ_{a_+} for positive connotation and σ_{a_-} for negative connotation using the following equations that take a linear combination of $f1$, $f2$, and $f3$ that we computed above with respect to each polarity.

$$\sigma_{a_+} = \alpha \times \sigma_{f1_+} + \beta \times \sigma_{f2_+} + \gamma \times \sigma_{f3_+} \quad (3)$$

$$\sigma_{a_-} = \alpha \times \sigma_{f1_-} + \beta \times \sigma_{f2_-} + \gamma \times \sigma_{f3_-} \quad (4)$$

Note that the coefficients α, β and γ are determined experimentally. We assign positive polarity to the word a, if $\sigma_{a_+} >> \sigma_{a_-}$ and vice versa.

7.1 Comparison against Sentiment Lexicon

The polarity defined in the connotation lexicon differs from that of conventional sentiment lexicons in which we aim to recognize more subtle sentiment that correlates with words. Nevertheless, we provide agreement statistics between our connotation lexicon and conventional sentiment lexicons for comparison purposes. We collect statistics with respect to the following two resources: General Inquirer (Stone and Hunt, 1963) and Opinion Finder (Wilson et al., 2005b).

For polarity $\lambda \in \{+, -\}$, let $count_{\text{sentlex}(\lambda)}$ denote the total number of words labeled as λ in a given sentiment lexicon, and let $count_{\text{agreement}(\lambda)}$ denote the total number of words labeled as λ by both the given sentiment lexicon and our connotation lexicon. In addition, let $count_{\text{overlap}(\lambda)}$ denote the total number of words that are labeled as λ by our connotation lexicon that are also included in the reference lexicon with or without the same polarity. Then we compute $prec_\lambda$ as follows:

$$prec_\lambda\ \% = \frac{count_{\text{agreement}(\lambda)}}{count_{\text{overlap}(\lambda)}} \times 100$$

We compare $prec_\lambda\ \%$ for three different segments of our lexicon: the top 100, top 1000, and the entire lexicon. We compare the lexicons provided by the seven variations of our algorithm. Results are shown in Table 3 & 4.

The acronym of each different method is defined as follows: **HITS-sT** & **HITS-aT** correspond to the **S**ymmetric (undirected) and **A**symmetric (directed) version of the **T**runcated method respectively. **HITS-sF** & **HITS-aF** correspond to the

Positive: include, offer, obtain, allow, build, increase, ensure, contain, pursue, fulfill, maintain, recommend, represent, require, respect
Negative: abate, die, condemn, deduce, investigate, commit, correct, apologize, debilitate, dispel, endure, exacerbate, indicate, induce, minimize

Table 5: Examples of newly discovered connotative predicates

Positive: boogie, housewarming, persuasiveness, kickoff, playhouse, diploma, intuitively, monument, inaugurate, troubleshooter, accompanist
Negative: seasickness, overleap, gangrenous, suppressing, fetishist, unspeakably, doubter, bloodmobile, bureaucratized

Table 6: Examples of newly discovered words with connotations: these words are treated as neutral in some conventional sentiment lexicons.

symmetric and asymmetric version of the **F**ocused method. Finally, **Page-aT** & **Page-aF** correspond to the **T**runcation and teleportation (**F**ocused) respectively.

Asymmetric HITS on a directed truncated graph (**HITS-aT**) and topic-sensitive PageRank (**Page-aF**) achieve the best performance in most cases, especially for top ranked words which have a higher average frequency. The difference between these two top performers is not large, but statistically significant using wilcoxon test with $p < 0.03$. Standard PageRank (**Page-aT**) achieves the third best performance overall. All these top performing ones (**HITS-aT**, **Page-aF**, **Page-aT**) outperform the baseline approach (**FREQ**) statistically significantly with $p < 0.001$. For brevity, we omit the PageRank results based on the undirected graphs, as the performance of those was not as good as that of directed ones.

7.2 Extrinsic Evaluation via Sentiment Analysis

Next we perform extrinsic evaluation to quantify the practical value of our connotation lexicon in concrete sentiment analysis applications. In particular, we make use of our connotation lexicon for binary sentiment classification tasks in two different ways:

- Unsupervised classification by voting. We define r as the ratio of positive polarity words to negative polarity words in the lexicon. In our experiment, penalty is 0 for positive and -0.5 for negative.

$$score(x_+) = 1 + penalty_+(r, \#positive)$$

$$score(x_-) = -1 + penalty_-(r, \#negative)$$

- Supervised classification using SVM. We use bag-of-words features for baseline. In order to quantify the effect of different lexicons, we add additional features based on the following scores as defined below:

$$score_{raw}(x) = \sum_{w \epsilon x} s(w)$$

$$score_{purity}(x) = \frac{score_{raw}(x)}{\sum_{w \epsilon x} abs(s(w))}$$

The two corpora we use are SemEval2007 (Strapparava and Mihalcea, 2007) and Sentiment Twitter.[1] The Twitter dataset consists of tweets containing either a *smiley* emoticon (representing positive sentiment) or a *frowny* emoticon (representing negative sentiment), we randomly select 50000 *smiley* tweets and 50000 *frowny* tweets.[2] We perform a 5-fold cross validation.

In Table 8, we find very promising results, particularly for Twitter dataset, which is known to be very noisy. Notice that the use of Top 6k words from our connotation lexicon along with OpinionFinder lexicon boost the performance up to 78.0%, which is significantly better than than 71.4% using only the conventional OpinionFinder lexicon. This result shows that our connotation lexicon nicely complements existing sentiment lexicon, improving practical sentiment analysis tasks.

[1] http://www.stanford.edu/~alecmgo/cs224n/twitterdata.2009.05.25.c.zip

[2] We filter out stop-words and words appearing less than 3 times. For Twitter, we also remove usernames of the format *@username* occurring within tweet bodies.

Algorithm	1st Round		2nd Round	
	Acc.	F-val	Acc.	F-val
Voting	68.7	65.4	71.0	68.5
Bag of Words	69.9	65.1	69.9	65.1
(〃) + OpFinder	74.7	75.0	74.7	75.0
BoW + Top 2k	73.3	74.5	73.7	75.4
(〃) + OpFinder	72.8	73.5	75.0	**77.6**
BoW + Top 6k	**76.6**	**77.1**	74.5	75.3
(〃) + OpFinder	74.1	73.5	**75.2**	76.0
BoW + Top 10k	74,1	73.5	74.2	73.8
(〃) + OpFinder	73.5	74.3	74.7	75.1

Table 7: SemEval Classification Result(%) — (〃) denotes that all features in the previous row are copied over.

Algorithm	1st Round		2nd Round	
	Acc.	F-val	Acc.	F-val
Voting	60.4	59.1	62.6	61.3
Bag of Words	69.9	72.1	69.9	72.1
(〃) + OpFinder	70.3	71.4	70.3	71.4
BoW + Top 2k	71.3	65.4	72.7	73.3
(〃) + OpFinder	69.4	63.1	73.1	74.6
BoW + Top 6k	**77.2**	69.0	76.4	77.6
(〃) + OpFinder	76.4	72.0	**76.8**	**78.0**
BoW + Top 10k	73.3	**73.5**	73.7	74.1
(〃) + OpFinder	74.1	69.5	73.5	74.2

Table 8: Twitter Classification Result(%) — (〃) denotes that all features in the previous row are copied over.

7.3 Intrinsic Evaluation via Human Judgment

In order to measure the quality of the connotation lexicon, we also perform human judgment study on a subset of the lexicon. Human judges are asked to quantify the degree of connotative polarity of each given word using an integer value between 1 and 5, where 1 and 5 correspond to the most negative and positive connotation respectively. When computing the annotator agreement score or evaluating our connotation lexicon against human judgment, we consolidate 1 and 2 into a single negative class and 4 and 5 into a single positive class. The Kappa score between two human annotators is 0.78.

As a control set, we also include 100 words taken from the General Inquirer lexicon: 50 words with positive sentiment, and 50 words with negative sentiment. These words are included so as to measure the quality of human judgment against a well-established sentiment lexicon. The words were presented in a random order so that the human judges will not know which words are from the General Inquirer lexicon and which are from our connotative lexicon. For the words in the control set, the annotators achieved 94% (97% lenient) accuracy on the positive set and 97% on the negative set.

Note that some words appear in both positive and negative connotation graphs, while others appear in only one of them. For instance, if a given word x appears as an argument for only positive connotative predicates, but never for negative ones, then x would appear only in the positive connotation graph. This means that for such word, we can assume the connotative polarity even without applying the algorithms for graph centrality. Therefore, we first evaluate the accuracy of the polarity of such words that appear only in one of the connotation graphs. We discard words with low frequency (300 in terms of Google n-gram frequency), and randomly select 50 words from each polarity. The accuracy of such words is 88% by strict evaluation and 94.5% by lenient evaluation, where lenient evaluation counts words in our polarized connotation lexicon to be correct if the human judges assign non-conflicting polarities, i.e., either neutral or identical polarity.

For words that appear in both positive and negative connotation graphs, we determine the final polarity of such words as one with higher scores given by HITS or PageRank. We randomly select words that rank at 5% of top 100, top 1000, top 2000, and top 5000 by each algorithm for human judgment. We only evaluate the top performing algorithms – HITS-aT and Page-aF – and FREQ baseline. The stratified performance for each of these methods is given in Table 9.

8 Related Work

Graph based approaches have been used in many previous research for lexicon induction. A technique named *label propagation* (Zhu and Ghahramani, 2002) has been used by Rao and Ravichandran (2009) and Velikovich et al. (2010), while random walk based approaches, PageRank in particular, have been used by Esuli and Sebastiani (2007). In our work, we explore the use of both HITS (Kleinberg, 1999) and PageRank (Page et al., 1999) and

Top #	Average		Positive		Negative	
	Str.	**Len.**	**Str.**	**Len.**	**Str.**	**Len.**
FREQ						
@100	73.5	87.3	72.2	91.1	74.7	83.5
@1000	51.8	78.6	44.4	75.6	81.8	90.9
@2000	66.9	74.7	73.1	84.2	57.3	60.0
@5000	61.5	81.3	61.4	84.1	62.0	70.0
HITS-aT						
@100	61.3	79.8	74.4	93.3	47.0	65.1
@1000	39.6	75.5	48.1	77.8	30.8	73.1
@2000	57.7	72.1	78.0	86.0	41.0	60.7
@5000	55.6	73.5	69.7	85.7	44.3	63.8
Page-aF						
@100	63.0	78.6	74.7	91.2	50.0	64.6
@1000	53.7	72.2	54.5	72.7	53.1	71.9
@2000	56.5	79.6	67.2	91.8	42.6	63.8
@5000	57.1	76.2	75.7	91.0	43.3	65.3

Table 9: Human Annotation Accuracies(%) – **Str.** denotes strict evaluation & **Len.** denotes lenient evaluation.

present systematic comparison of various options for graph representation and encoding of prior knowledge. We are not aware of any previous research that made use of HITS algorithm for connotation or sentiment lexicon induction.

Much of previous research investigated the use of dictionary network (e.g., WordNet) for lexicon induction (e.g., Kamps et al. (2004), Takamura et al. (2005), Adreevskaia and Bergler (2006), Esuli and Sebastiani (2006), Su and Markert (2009), Mohammad et al. (2009)), while relatively less research investigated the use of web documents (e.g., Kaji and Kitsuregawa (2007), Velikovich et al. (2010))).

Wilson et al. (2005b) first introduced the sentiment lexicon, spawning a great deal of research thereafter. At the beginning, sentiment lexicons were designed to include only those words that *express* sentiment, that is, *subjective* words. However in recent years, sentiment lexicons started expanding to include some of those words that simply associate with sentiment, even if those words are purely objective (e.g., Velikovich et al. (2010), Baccianella et al. (2010)). This trend applies even to the most recent version of the lexicon of Wilson et al. (2005b). We conjecture that this trend of broader coverage suggests that such lexicons are practically more useful than sentiment lexicons that include only those words that are strictly subjective. In this work, we make this transition more explicit and intentional, by introducing a novel *connotation lexicon*.

Mohammad and Turney (2010) focussed on emotion *evoked* by common words and phrases. The spirit of their work shares some similarity with ours in that it aims to find the emotion *evoked* by words, as opposed to *expressed*. Two main differences are: (1) our work aims to discover even more subtle association of words with sentiment, and (2) we present a nearly unsupervised approach, while Mohammad and Turney (2010) explored the use of Mechanical Turk to build the lexicon based on human judgment.

In the work of Osgood et al. (1957), it has been discussed that connotative meaning of words can be measured in multiple scales of semantic differential, for example, the degree of "goodness" and "badness". Our work presents statistical approaches that measure one such semantic differential automatically. Our graph construction to capture word-to-word relation is analogous to that of Collins-Thompson and Callan (2007), where the graph representation was used to model more general definitions of words.

9 Conclusion

We introduced the *connotation lexicon*, a novel lexicon that list words with connotative polarity, which will be made publically available. We also presented graph-based algorithms for learning connotation lexicon together with connotative predicates in a nearly unsupervised manner. Our approaches are grounded on the linguistic insight with respect to the selectional preference of connotative predicates. Empirical study demonstrates the practical value of the connotation lexicon for sentiment analysis encouraging further research in this direction.

Acknowledgments

We wholeheartedly thank the reviewers for very helpful and insightful comments.

References

Alina Adreevskaia and Sabine Bergler. 2006. Mining wordnet for fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 209–216.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Monica Bianchini, Marco Gori, and Franco Scarselli. 2005. Inside pagerank. *ACM Trans. Internet Technol.*, 5:92–128, February.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1. In *Linguistic Data Consortium, ISBN: 1-58563-397-6, Philadelphia.*

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16:22–29, March.

K. Collins-Thompson and J. Callan. 2007. Automatic and human scoring of word definition responses. In *Proceedings of NAACL HLT*, pages 476–483.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*, pages 417–422.

Andrea Esuli and Fabrizio Sebastiani. 2007. Pageranking wordnet synsets: An application to opinion mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 424–431. Association for Computational Linguistics.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the Eleventh International World Wide Web Conference*, Honolulu, Hawaii.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of HTML documents. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1075–1083.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten De Rijke. 2004. Using wordnet to measure semantic orientation of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 1115–1118.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *JOURNAL OF THE ACM*, 46(5):604–632.

B. Louw, M. Baker, G. Francis, and E. Tognini-Bonelli. 1993. Irony in the text or insincerity in the writer? the diagnostic potential of semantic prosodies. *TEXT AND TECHNOLOGY IN HONOUR OF JOHN SINCLAIR*, pages 157–176.

Saif Mohammad and Peter Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34, Los Angeles, CA, June. Association for Computational Linguistics.

Saif Mohammad, Cody Dunne, and Bonnie Dorr. 2009. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 599–608, Singapore, August. Association for Computational Linguistics.

C. E. Osgood, G. Suci, and P. Tannenbaum. 1957. *The measurement of meaning*. University of Illinois Press, Urbana, IL.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682, Morristown, NJ, USA. Association for Computational Linguistics.

John Sinclair. 1991. *Corpus, concordance, collocation*. Describing English language. Oxford University Press.

A. Stefanowitsch and S.T. Gries. 2003. Collostructions: Investigating the interaction of words and constructions. *International Journal of Corpus Linguistics*, 8(2):209–243.

Philip J. Stone and Earl B. Hunt. 1963. A computer approach to content analysis: studies using the general inquirer system. In *Proceedings of the May 21-23, 1963, spring joint computer conference*, AFIPS '63 (Spring), pages 241–256, New York, NY, USA. ACM.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: affective text. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74, Morristown, NJ, USA. Association for Computational Linguistics.

M. Stubbs. 1995. Collocations and semantic profiles: on the cause of the trouble with quantitative studies. *Functions of language*, 2(1):23–55.

Fangzhong Su and Katja Markert. 2009. Subjectivity recognition on word senses via semi-supervised mincuts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1–9. Association for Computational Linguistics.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of ACL-05, 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, US. Association for Computational Linguistics.

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

Peter Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation (formerly Computers and the Humanities)*, 39(2/3):164–210.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005a. Opinionfinder: a system for subjectivity analysis. In *Proceedings of HLT/EMNLP on Interactive Demonstrations*, pages 34–35, Morristown, NJ, USA. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005b. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354, Morristown, NJ, USA. Association for Computational Linguistics.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. In *Technical Report CMU-CALD-02-107*. CarnegieMellon University.

Hypotheses Selection Criteria in a Reranking Framework for Spoken Language Understanding

Marco Dinarelli
LIMSI-CNRS
B.P. 133, 91403 Orsay Cedex
France
marcod@limsi.fr

Sophie Rosset
LIMSI-CNRS
B.P. 133, 91403 Orsay Cedex
France
rosset@limsi.fr

Abstract

Reranking models have been successfully applied to many tasks of Natural Language Processing. However, there are two aspects of this approach that need a deeper investigation: (i) Assessment of hypotheses generated for reranking at classification phase: baseline models generate a list of hypotheses and these are used for reranking without any assessment; (ii) Detection of cases where reranking models provide a worst result: the best hypothesis provided by the reranking model is assumed to be always the best result. In some cases the reranking model provides an incorrect hypothesis while the baseline best hypothesis is correct, especially when baseline models are accurate. In this paper we propose solutions for these two aspects: (i) a semantic inconsistency metric to select possibly more correct n-best hypotheses, from a large set generated by an SLU basiline model. The selected hypotheses are reranked applying a state-of-the-art model based on Partial Tree Kernels, which encode SLU hypotheses in Support Vector Machines with complex structured features; (ii) finally, we apply a decision strategy, based on confidence values, to select the final hypothesis between the first ranked hypothesis provided by the baseline SLU model and the first ranked hypothesis provided by the re-ranker. We show the effectiveness of these solutions presenting comparative results obtained reranking hypotheses generated by a very accurate Conditional Random Field model. We evaluate our approach on the French MEDIA corpus. The results show significant improvements with respect to current state-of-the-art and previous re-ranking models.

1 Introduction

Discriminative reranking is a widely used approach for several Natural Language Processing (NLP) tasks: Syntactic Parsing (Collins, 2000), Named Entity Recognition (Collins, 2000; Collins and Duffy, 2001), Semantic Role Labelling (Moschitti et al., 2008), Machine Translation (Shen et al., 2004), Question Answering (Moschitti et al., 2007). Recently reranking approaches have been successfully applied also to Spoken Language Understanding (SLU) (Dinarelli et al., 2009b).

Discriminative Reranking combines two models: a first SLU model is used to generate a ranked list of *n*-best hypotheses; a reranking model sorts the list based on a different score and the final result is the new top ranked hypothesis. The advantage of reranking approaches is in the possibility to learn directly complex dependencies in the output domain, as this is provided in the hypotheses generated by the baseline model.

In previous approaches complex features are extracted from the hypotheses for both training and classification phase, but there are very few studies on approaches that can be applied to search in the hypotheses space generated by the baseline SLU model. Moreover, to keep overall computational cost reasonable, the size of the n-best list is typically small (few tens). This is a limitation since the larger is the hypotheses space generated, the more likely is to find a better hypothesis. On the other hand, reranking a large set of hypotheses is computationally

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1104–1115,
Edinburgh, Scotland, UK, July 27–31, 2011.

expensive, thus a strategy to select the best hypotheses to be re-ranked would overcome this problem.

Another aspect of reranking that deserves to be deeper studied is its applicability. Although a reranking model improves the baseline model in the overall performance, in some cases the reranked best hypotheses can contain more mistakes than the baseline best hypothesis. A strategy to decide when the reranking model should be applied and when the first hypothesis of the baseline model is more accurate would improve reranking performances.

In this paper, we propose two new models for improving discriminative reranking: (a) a semantic inconsistency metric that can be applied to SLU hypotheses to select those that are more likely to be correct; (b) a model selection strategy based on the confidence scores provided by the baseline SLU model and the reranker. This provides a decision function that detects if the original top ranked hypothesis is more accurate than the reranked best hypothesis.

Our re-ranking strategies turn out to be effective on very accurate baseline models based on state-of-the-art Conditinal Random Fields (CRF) implementation (Lavergne et al., 2010). We evaluate our approach on the well-known French MEDIA corpus for SLU (Bonneau-Maynard et al., 2006). The results show that our approach significantly improves both "traditional" reranking approaches and state-of-the-art SLU models.

The remainder of the paper is organized as follows: in Section 2 we introduce the SLU task. Section 3 describes our discriminative reranking framework for SLU, in particular the baseline model adopted, in sub-section 3.1, and the reranking model, in sub-section 3.2. Section 4 describes the two strategies proposed in this paper for SLU reranking, whereas the experiments to evaluate our approaches are described in Section 5. Finally, after a discussion in Section 6, in Section 7 we draw some conclusions.

2 Spoken Language Understanding

Spoken Language Understanding is the task of representing and extracting the meaning of natural language sentences. Designing a general meaning representation which can capture the semantics of a spoken language is very complex. Therefore, in practice, the meaning representations depend on the specific application domain being modeled.

For the corpus used in this work, the semantic representation is defined in an ontology described in (Bonneau-Maynard et al., 2006). As an example, given the following natural language sentence translated from the MEDIA corpus:

"*Good morning I would like to book an hotel room in London*"

The semantic representation extraction for the SLU task is performed in two steps:

1. **Automatic Concept Labeling**

   Null{*Good morning*} **command-task**{*I would like to book*} **object-bd**{*an hotel room*} **localization-city**{*in London*}

2. **Attribute-Value Extraction**

   command-task[reservation] **object-bd**[hotel] **localization-city**[London]

command-task, **object-bd** and **localization-city** are three domain concepts, called also "attributes", defined in the ontology and **Null** is the concept for words not associated to any concept. As shown in the example, **Null** concepts are removed from the final output since they don't bring any semantic content with respect to the application domain. **reservation**, **hotel** and **London** are the normalized attribute values, defined also in the application ontology. This representation is usually called attribute-value representation.

In the last decade several probabilistic models have been proposed for the Automatic Concept Labeling step: in (Raymond et al., 2006) a conceptual language model encoded in Stochastic Finite State Transducers (SFST) is proposed. In (Raymond and Riccardi, 2007), the SFST-based model is compared with Support Vector Machines (SVM) (Vapnik, 1998) and Conditional Random Fields (CRF) (Lafferty et al., 2001). Moreover, in (Hahn et al., 2008a) two more models are applied to SLU: a Maximum Entropy (EM) model and a model coming from the Statistical Machine Translation (SMT) community (it is actually a log-linear combination of SMT models). Among these models, CRF has shown in general superior performances on sequence labeling tasks like Named Entity Recognition (NER) (Tjong Kim Sang and De Meulder, 2003), Grapheme-to-Phoneme transcription (Sejnowski and Rosenberg,

1987) and also Spoken Language Understanding (Hahn et al., 2008a).

In addition to individual systems, more recently also some system combination approaches have been tried on SLU. In (Hahn et al., 2010), two such approaches are compared, one based on weighted ROVER (Fiscus, 1997) while the other is the reranking approach proposed in (Dinarelli et al., 2009b). Both system combination approaches are applied on the MEDIA corpus, thus we will refer to (Hahn et al., 2010) for a comparison with our approach.

Like the other tasks mentioned above, SLU is usually a supervised learning task, this means that models are learned from annotated data. This is an important aspect to take into account when designing SLU systems. In this respect accurate SLU models can in part alleviate the problem of manually annotating data.

The second step of SLU, that is Attribute Value Extraction (from now on *AVE*) is performed with two approaches: a) Rule-based approaches apply Regular Expressions (RE) to map the words realizing a concept into a normalized value. Regular expressions are defined for each attribute-value pair. Given a concept and its realizing surface form, if a RE for that concept matches the surface, the corresponding value is returned.

An example of surfaces that can be mapped into the value **hotel** given the concept **object-bd** is:

1. *an hotel room*
2. *a hotel room*
3. *the hotel*

   ...

Note that these surfaces share the same keyword for the concept **object-bd**, which is "*hotel*". Thus, a possible rule extracted from data, for the concept **object-bd** can be:

$R_{object-bd}(S)$ =
if S = "an hotel room" **or**
S = "a hotel room" **or**
S = "the hotel" **then**
return "hotel"
end

This kind of rules can be easily refined using regular expressions, so that they can capture all possible linguistic patterns containing the triggering keyword ("hotel" in the example).

b) The other approach used for attribute value extraction is based on probabilistic models. In this case the model learns from data the conditional probability of values V, given the concept C and the corresponding sequence of words W realizing the concept: $P(V|W, C)$.

The most meaningful work about AVE approaches in SLU tasks is (Hahn et al., 2010).

The model used in this work for *Automatic Concept Labeling* is based on CRF. For the *Attribute-Value Extraction* phase we use a combination of rule based and probabilistic approaches. The first is made of regular expressions, as explained above. The probabilistic approach is based again on CRF.

3 Reranking Framework

This section describes the different models involved in the pipeline realising our reranking framework:

- Conditional Random Fields
- Semantic Inconsistency Metric for hypotheses selection, which is optional and is applied only at the classification phase
- Support Vector Machines with Partial Tree Kernel
- Decision Strategy to detect when the top ranked hypothesis of the baseline model is more accurate than the reranked best hypothesis

It is important to underline that the phases involved in the reranking framewrok are distinguished for a matter of clarity. In principle, the phases from the hypotheses selection to the last, the decision strategy, can be thought of as a whole reranking model.

In the next two subsection we describe the two models used for hypotheses generation and for reranking: CRF and SVM with kernel methods. The two improvements proposed in this paper and listed above are presented in a dedicated section (4).

3.1 Conditional Random Fields

CRFs have been proposed for the first time for sequence segmentation and labeling tasks in (Lafferty et al., 2001). This model belongs to the family of exponential or log-linear models. Its main characteristics are the possibility to include a huge number

of features, like the Maximum Entropy (ME) model, but computing global conditional probabilities normalized at sentence level, instead of position level like in ME. In particular this last point results very effective since it solves the label bias problem, as pointed out in (Lafferty et al., 2001).

Given a sequence of N words $W_1^N = w_1, ..., w_N$ and its corresponding sequence of concepts $C_1^N = c_1, ..., c_N$, CRF trains the conditional probabilities

$$P(C_1^N|W_1^N) = \frac{1}{Z}\prod_{n=1}^{N}\exp\left(\sum_{m=1}^{M}\lambda_m \cdot h_m(c_{n-1}, c_n, w_{n-2}^{n+2})\right) \quad (1)$$

where λ_m are the training parameters. $h_m(c_{n-1}, c_n, w_{n-2}^{n+2})$ are the feature functions capturing conditional dependencies of concepts and words. Z is a probability normalization factor in order to model well defined probability distribution:

$$Z = \sum_{\tilde{c}_1^N}\prod_{n=1}^{N} H(\tilde{c}_{n-1}, \tilde{c}_n, w_{n-2}^{n+2}) \quad (2)$$

here $\tilde{c}_{n-1}$ and $\tilde{c}_n$ are the concepts hypothesized for the previous and current words, $H(\tilde{c}_{n-1}, \tilde{c}_n, w_{n-2}^{n+2})$ is an abbreviation for $\sum_{m=1}^{M}\lambda_m \cdot h_m(c_{n-1}, c_n, w_{n-2}^{n+2})$.

The CRF model used for the *Attribute-Value Extraction* phase learns in the same way the conditional probability $P(V_1^N|C_1^N, W_1^N)$. In particular we use attributes-words concatenations on the source side and attribute values on the target side.

Two particular effective implementations of CRFs have been recently proposed. One is described in (Hahn et al., 2009) and uses a margin based criterion for probabilities estimation. The other is described in (Lavergne et al., 2010) and has been implemented in the software *wapiti*[1]. The latter solution in particular trains the model using two different regularization factors at the same time:

Gaussian prior, used as *l2* regularization and used in many softwares to avoid overfitting;

Laplacian prior, used as *l1* regularization (Riezler and Vasserman, 2010), which has the effect to filter out features with very low scores.

[1]available at `http://wapiti.limsi.fr`

The two regularization parameters are used together in the model implementing the so-called *elastic net* regularization (Zou and Hastie, 2005):

$$l(\lambda) + \rho_1\|\lambda\|_1 + \frac{\rho_2}{2}\|\lambda\|_2^2 \quad (3)$$

λ is the set of parameters of the model introduced in equation 1, $l(\lambda)$ is the minus-logarithm of equation 1, used as loss function for training CRF. $\|\lambda\|_1$ and $\|\lambda\|_2$ are the *l1* and *l2* regularization, respectively, while ρ_1 and ρ_2 are two parameters that can be optimized as usual on development data or with cross validation.

As explained in (Lavergne et al., 2010), using *l1* regularization is an effective way for feature selection in CRF at training time. Note that other approaches have been proposed for feature selection, e.g. in (McCallum, 2003). This type of features selection, performed directly at training time, yields very accurate models, since only the most meaningful features are kept in the final model, which guarantee a strong robustness on unseen data.

In this work we refer in particular to the CRF implementation described in (Lavergne et al., 2010).

3.2 SVM and Kernel Methods

Our reranking model is based on SVM (Vapnik, 1998) with the use of the Partial Tree Kernel defined in (Moschitti, 2006).

SVMs are well-known machine learning algorithms belonging to the class of maximal-margin linear classifiers (Vapnik, 1998). The model represents a hyperplane which separates the training examples with a maximum margin. The hyperplane is learned using optimization theory and is represented in the dual form as a linear combination of training examples:

$\sum_{i=1..l} y_i\alpha_i\vec{x_i}\vec{x} + b = 0$,

where $\vec{x_i}, i \in [1, .., l]$ are training examples representing objects o_i and o in any feature space, y_i is the label associated with $\vec{x_i}$ and α_i are the lagrange multipliers. The dual form of the hyperplane shows that SVM training depends on the inner product between instances. Kernel methods theory (Shawe-Taylor and Cristianini, 2004), allows us to substitute the inner product with a so-called kernel function, computing the same result: $K(o_i, o) = \vec{x_i} \cdot \vec{x}$.

The interesting aspect of using such formulation is the possibility to compare objects in arbitrarily complex feature spaces implicitly, i.e. without knowing the features to be used. Since in real world scenarios data cannot be classified using a simple linear classifier, kernel methods can be used to carry out learning in complex feature spaces. In this work we use the Partial Tree Kernel (PTK) (Moschitti, 2006).

3.3 Reranking Model

In order to give an effective representation to SLU hypotheses in SVM, since we are using PTK, we need to represent as trees SLU hypotheses like the one described in section 2.

This problem is easily solved by transforming the hypotheses into trees like the one depicted in figure 1. Although there may be more formal solutions to represent semantic information of SLU hypotheses as trees, we would like to remark that the tree structure shown in figure 1 contains all the key information needed for our purposes: the first level of the tree represents the concept sequence annotated on surface form. The second level of the tree allow to implicitly represent the segmentation of each concept, while the third level, i.e. the leaves, are the input words. Moreover, from figure 1 we removed word categories associated to words in order to keep the figure readable. Word categories are provided together with the corpus as an application knowledge base. They comprise domain categories like city names, hotel names, street names etc., and some domain independent categories like numbers, dates, months etc. The categories are used at the same level of words, they provide a generalization over words and alleviate the effect of Out-of-Vocabulary (OOV) words.

The CRF model used as baseline generates the *n* most likely conceptual annotations for each input sentence. These are ranked by the global conditional probability of the concept sequence, given the input word sequence of CRF. The *n*-best list produced by the baseline model is the list of candidate hypotheses $H_1, H_2, .., H_n$ used in the reranking step.

The candidate hypotheses are organized into pairs, e.g. (H_1, H_2) or (H_1, H_3). We build training pairs such that a reranker can learn to select the best one between the two hypotheses in a pair, i.e. the more correct hypothesis with respect to a reference annotation and a given metric. In particular, we compute the edit distance of each hypothesis in the list, with respect to the manual annotation taken from the corpus. The best hypothesis H_b is used to build positive instances for the reranker as pairs (H_b, H_i) for $i \in [1..n]$ and $i \neq b$, negative instances are built as (H_i, H_b), with same constraints on index i. This means that, if n hypotheses are generated for a sentence, $2 \cdot n$ instances are generated from them. Note that by construction of pairs the model is symmetric, this provides a property that will be exploited at classification phase, as described in (Shen et al., 2003b).

Hypotheses are then converted into trees like the one shown in figure 1. Pairs of trees $e_k = (t_{i,k}, t_{j,k})$, for k varying along all the training or classification instances, are given as input to the SVM model to train the reranker using the following reranking kernel:

$$\begin{aligned} K_R(e_1, e_2) &= PTK(t_{1,1}, t_{1,2}) + PTK(t_{2,1}, t_{2,2}) \\ &- PTK(t_{1,1}, t_{2,2}) - PTK(t_{2,1}, t_{1,2}), \end{aligned} \quad (4)$$

where e_1 and e_2 are two pairs of trees to be compared.

The reranking kernel in equation 4, consisting in summing four different kernels, has been proposed in (Shen et al., 2003b) for syntactic parsing reranking, where the basic kernel was a Tree Kernel, and the idea was taken in turn from (Heibrich et al., 2000), where pairs where used to learn preference ranking. The same idea appears also, in a slightly different form, in early work about reranking, e.g. (Collins and Duffy, 2002). The same reranking schema has been used also in (Shen et al., 2004) for reranking different candidate hypotheses for machine translation.

For classification, observing that the model is symmetric and exploiting kernel properties, we can use, as classification instances, simple hypotheses instead of pairs. More precisely we use pairs where the second hypothesis is empty, i.e. $(H_i, 0)$, for $i \in [1..n]$. This simplification allow a relatively fast classification phase, since only n instances are generated for each sentence, instead of n^2. This simplification has been proposed in (Shen et al., 2003b).

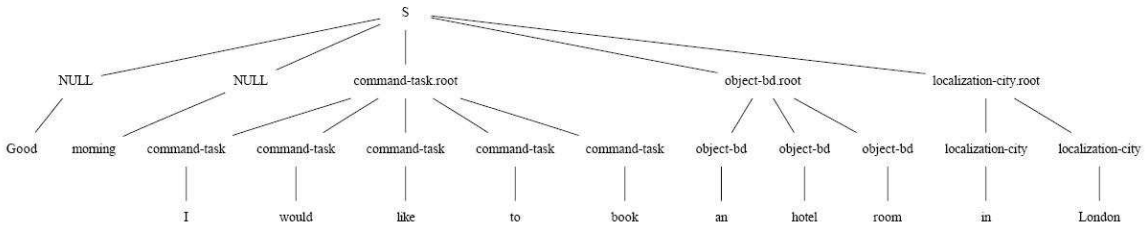


Figure 1: An example of semantic tree constructed from an SLU hypothesis from the MEDIA corpus and used in PTK

4 Hypotheses Selection Criteria

This section describes the main contribution of our work: first, a semantic inconsistency metric based on the AVE phase of SLU and allowing to select hypotheses generated by the baseline model; second, a strategy to decide, after the reranking phase, if it is more likely that the baseline best hypothesis is more accurate than the best reranked hypothesis and allowing to recover the mistake. Similar ideas have been proposed in (Dinarelli et al., 2010), here we propose a significant evolution and we give a much wider description and evaluation.

4.1 Hypotheses Selection via Attribute Value Extraction (AVE)

In previous reranking approaches (Collins, 2000; Collins and Duffy, 2002; Shen et al., 2003a; Shen et al., 2003b; Shen et al., 2004; Collins and Koo, 2005; Kudo et al., 2005; Dinarelli et al., 2009b), few hypotheses are generated with the baseline model, ranked by the model probability. These are then used for the reranking model. An interesting strategy to improve reranking performance is the selection of the best set of hypotheses to be reranked.

In this work we propose a semantic inconsistency metric (SIM) based on the attribute-value extraction phase that allows to select better n-best hypotheses. We combine the scores provided by the rule based approach and the CRF approach for AVE, computing a confidence measure.

The rule-based approach for AVE is defined by a set of rules that map concepts and their realizing words into the corresponding value. The rules are extracted from the training data, thus they are defined to extract correct values from well formed phrases annotated with correct concepts. This means that when the corresponding words are annotated with a wrong concept, the extracted value will probably be wrong. We use this property to compute a semantic inconsistency value for hypotheses, which in turn allows to select hypotheses with higher probabilities to be correct.

We show the application of SIM using the same example of Section 2. For space issues we abbreviate **command-task** with **com-task**, **object-bd** with **obj-bd** and **localization-city** with **loc-city**. We also suppose to have already removed **Null** concepts. From the same sentence, the three first hypotheses that may be generated by the baseline model are:

1. **obj-bd**{*I would like to book*} **obj-bd**{*an hotel room*} **loc-city**{*in London*}
2. **com-task**{*I would like to book*} **obj-bd**{*an hotel room*} **loc-city**{*in London*}
3. **com-task**{*I would like to book*} **obj-bd**{*an hotel*} **obj-bd**{*room*} **loc-city**{*in London*}

Two of these annotations show typical errors of an SLU model:
(i) wrong concepts annotation: in the first hypothesis the phrase "I would like to book" is erroneously annotated as **obj-bd**;
(ii) wrong concept segmentation: in the third hypothesis the phrase "an hotel room" is split in two concepts.

If we apply the AVE module to these hypotheses the result is:

1. **obj-bd**[] **obj-bd**[hotel] **loc-city**[london]
2. **cmd-task**[reservation] **obj-bd**[hotel] **loc-city**[london]
3. **cmd-task**[reservation] **obj-bd**[hotel] **obj-bd**[] **loc-city**[london]

As we can see the first concept **obj-bd** in the first hypothesis has an empty value since it was incorrectly annotated and, therefore, it is not supported

MEDIA	training		dev		test	
# sentences	12,908		1,259		3,005	
	words	concepts	words	concepts	words	concepts
# tokens	94,466	43,078	10,849	4,705	25,606	11,383
# vocabulary	2,210	99	838	66	1,276	78
# singletons	798	16	338	4	494	10
# OOV rate [%]	–	–	1.33	0.02	1.39	0.04

Table 1: Statistics of the MEDIA training and evaluation sets used for all experiments.

by words from which the AVE module can extract a correct value. In this case, the output of AVE is empty. In the same way, in the third hypothesis, the AVE module cannot extract a correct value from the phrase "room" since it doesn't contain any keyword for a **obj-bd** concept.

For each hypothesis, our SIM simply counts the number of wrong (or empty) values. In the example above, we have 1, 0 and 1 for the three hypothesis, respectively. Accordingly, the most accurate hypothesis under SIM is the second, which is also the correct one.

In order to combine the SIM score computed by the rule-based AVE module with the score provided by the CRF AVE model, we consider per-concept scores from both approaches. In particular, we formalize the definition of the SIM metric above on a concept c_i as $SIM(c_i, w_i^{1,...,m})$. The value of SIM is simply 0 if the rule-based AVE module can extract a value from the surface form $w_i^{1,...,m}$ realizing the concept c_i. 1 otherwise. For each concept in a hypothesis, we compute its semantic consistency $s(c_i)$ as

$$s(c_i) = \frac{P(v_i|c_i, w_i^{1,...,m})}{SIM(c_i, w_i^{1,...,m}) + 1} \quad (5)$$

where $P(v_i|c_i, w_i^{1,...,m})$ is the conditional probability output by the CRF model for the value v_i, given the concept c_i and its realizing surface $w_i^{1,...,m}$. Equation 5 means that the CRF score provided for a given value is halved if SIM returns 1, i.e. if the AVE module cannot extract any value. Otherwise the score output by the CRF AVE model is kept unchanged. The semantic inconsistency metric of an hypothesis H_k containing the concept sequence $C_1^N = c_1, ..., c_N$ is then defined as

$$S(H_k) = \sum_{i=1}^{N} s(c_i) \quad (6)$$

Using $S(H_k)$ as semantic inconsistency metric, we generate a huge number of hypotheses with the baseline model and we select only the top *n*-best. We use these hypotheses in the discriminative reranking model, instead of the original *n*-best generated by the CRF model. For simplicity, in general context we denote $S(H_k)$ as `SIM`.

4.2 Wrong Rerank Rejection

After the reranking model is applied, the first hypothesis is selected as final result. This choice assumes that the new hypothesis is more accurate than the one provided by the baseline model. In general this assumption is not true. Indeed, a reranking model must be carefully tuned in order to correctly rerank wrong first best hypotheses but keeping the original baseline best for correct hypotheses. When the baseline model is relatively accurate, the latter case occurs in most of the cases. In this situation it becomes hard to train an accurate reranking model.

Our idea to overcome this problem is to apply the reranking model and then post-process results to detect when the original best hypothesis is actually better than the reranked best.

For this purpose we propose a simple strategy based on the scores computed by the two models involved in reranking: CRF for the baseline and SVM with PTK for reranking.

Let H_{crf} and H_{RR} be the best hypothesis of the CRF and reranking (RR) models, respectively. Let $S_{crf}(H_{crf})$ and $S_{crf}(H_{RR})$ be the scores of the CRF model for H_{crf} and H_{RR}. In the same way, let $S_{RR}(H_{crf})$ and $S_{RR}(H_{RR})$ be the scores of the reranking model on the same hypotheses. We define the *confidence margin* of the CRF model the quantity: $M_{crf} = S_{crf}(H_{crf}) - S_{crf}(H_{RR})$.

In the same way we define the *confidence margin* of the RR model: $M_{RR} = S_{RR}(H_{crf}) - S_{RR}(H_{RR})$.

We compute two thresholds T_{crf} and T_{RR} for the

Average score	Feature type
0.0528186	Pref2
0.044189	CATEGORY-2
0.0355579	CATEGORY
0.0354006	Pref3-2
0.0338949	Pref4-2
0.0332647	Suff3-2
0.0314831	Suff2
0.030613	Suff4-2
...	...
0.0165602	Suff1
0.000579602	Pref1

Table 2: Ranks of average score given by the CRF model to feature types

two margins with respect to error rate minimization (with a *"line search"* algorithm).

We select the final best interpretation hypothesis for a given sentence with the decision function:

$$BestHypothesis = \begin{cases} H_{RR} & \text{if } M_{crf} \leq T_{crf} \text{ and } M_{RR} \geq T_{RR} \\ H_{crf} & \text{otherwise.} \end{cases}$$

Since this strategy allows to recover from reranking mistakes, we call it Wrong Rerank Rejection (WRR).

5 Experiments

The data used in our experiments are taken from the French MEDIA corpus (Bonneau-Maynard et al., 2006). The corpus is made of 1.250 Human-Machine dialogs acquired with a Wizard-of-Oz approach in the domain of informtation and reservation of French hotels. The data are split into training, development and test set. Statistics of the corpus are presented in table 1.

For our CRF models, both *Automatic Concept Annotation* and *Attribute Value Extraction* SLU phases, we used wapiti[2] (Lavergne et al., 2010). The CRF model for the first SLU phase integrates a traditional set of features like word prefixes and suffixes (of length up to 5), plus some *Yes/No* features like *"Does the word start with capital letter ?"*, *"Does the word contain non alphanumeric characters ?"*, *"Is the word preceded by non alphanumeric characteris ?"* etc. The CRF model for AVE integrates only words, prefixes and suffixes (length 3 and 4) concatenated with concepts. Since in this case labels are attribute values, which are a huge set with respect to concepts ($\tilde{7}$00 VS 99), using a lot of features would make model training problematic. Despite the reduced set of features, training error rate at both token and sentence level is under 1%. We didn't carry out optimization for parameters ρ_1 and ρ_2 of the elastic net (see section 3.1), default values lead in most cases to very accurate models.

MEDIA Text Input	DEV		TEST	
Model	Attr	Attr+Val	Attr	Attr+Val
CRF	12.1%	14.8%	11.5%	13.8%
CRF+RR	12.0%	14.6%	11.5%	13.7%
CRF+RR_{SIM}	11.7%	13.9%	11.3%	13.4%
CRF+RR_{WRR}	11.2%	13.4%	11.3%	13.0%

Table 3: Results of baseline CRF model and reranking models on MEDIA text input

Reranking models based on SVM and PTK have been trained with *"SVM-Light-TK"*[3]. Kernel parameters *M* and SVM parameter *C* have been optimized on the development set, as well as thresholds for the WRR (see section 4.2).

Concerning hypotheses generation, for training we generate 100 hypotheses, we select the best with respect to the edit distance and the reference annotation and we keep a total of 10 hypotheses to build pairs. For classification, with the "standard" reranking approach we generate and we keep the 10 best hypotheses. While using SIM for hypotheses selection, we generate 1.000 hypotheses and we keep the 10 best with respect to SIM. 1.000 is the best threshold between oracle accuracy and computational cost for evaluating the hypotheses.

Experiments have been performed on both manual and automatic transcriptions of dialog turns. For automatic transcriptions the WER of the ASR is 30.3% on development set and 31.4% on test set.

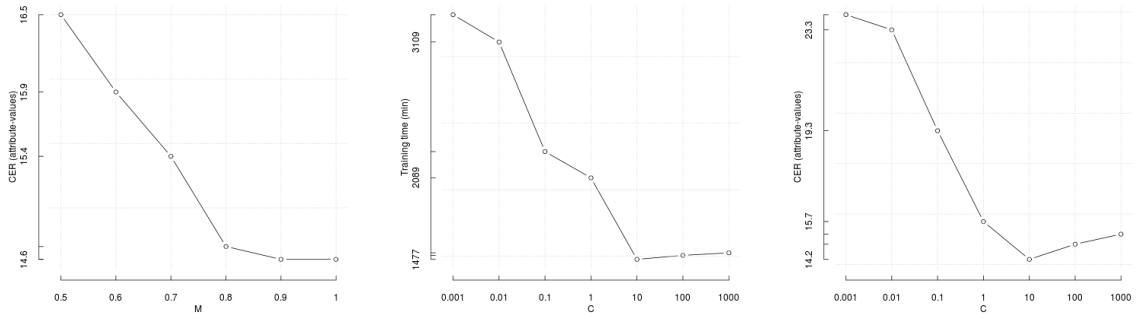
All results are reported in terms of Concept Error Rate (CER), which is the same as WER, but it is computed on concept sequences. In all cases we give results for both attributes only and attributes and values extraction

5.1 Results

In order to understand feature relevance, in table 2 we report feature types ranked by the average score given by the CRF model. Each type correspond to features at any position with respect to the target

[2]available at http://wapiti.limsi.fr

[3]available at http://disi.unitn.it/moschitti/Tree-Kernel.htm



(a) M kernel parameter VS CER (on attribute-value extraction)

(b) C SVM parameter VS training time

(c) C SVM parameter VS CER (on attribute-value extraction)

Figure 2: Optimization of the PTK M parameter and C parameter of SVM

MEDIA Speech Input	DEV		TEST	
Model	Attr	Attr+Val	Attr	Attr+Val
CRF	24.1%	29.1%	23.7%	27.6%
CRF+RR	23.9%	29.1%	23.5%	27.6%
CRF+RR_{SIM}	23.9%	28.3%	23.2%	26.8%
CRF+RR_{WRR}	23.3%	27.5%	22.7%	26.1%

Table 4: Results of baseline CRF model and reranking models on MEDIA speech input

word, with label unigrams. In contrast observation unigrams are distinguished from bigrams using suffixes *-1* and *-2* respectively. Feature types *wrd* are words converted to lower case, *Wrd* are words kept with original capitalization. Feature types *Pre*n are word prefixes of length n, *Suf*n are word suffixes of length n. *CATEGORY* features are word categories (see section 3.3). As we can see from the table, although feature relevance depends of course from the task, surprisingly word prefixes of length 2 are the most meaningful features. As expected, *CATEGORY* features are also very relevant features, since they provide a strong generalization over words. Another expected outcome is the fact that prefixes and suffixes of length 1 are the least relevant features.

In figure 2(a), 2(b) and 2(c) we show the curves resulting from optimization of parameters of reranking models. In particular we optimized the *M* kernel parameter (μ decay factor, see (Moschitti, 2006) for details), and the *C* SVM parameter, i.e. the scale factor for the soft margin (please refer to (Vapnik, 1998) for SVM details). Figure 2(b) shows the learning time as a function of the *C* SVM parameter. This gives an idea of how long takes training our reranking models.

In table 3 and 4 we report comparative results over the baseline CRF model, the baseline reranking model (*CRF+RR*) and the reranking models obtained applying the two improvements proposed in this work (*CRF+*RR_{SIM} and *CRF+*RR_{WRR}). As we can see, the baseline reranking model does not improve significantly the baseline CRF model. This outcome is expected since we don't use any other information in the reranking model than the semantic tree shown in figure 1. Previous approaches like for example (Collins and Duffy, 2002), use the baseline model score as feature, as that the reranking model cannot do worst than the baseline model. As we pointed out in section 4.2, this solution require a fine tuning of the reranking model, especially when the baseline model is relatively accurate. In our case, the CRF model has a Sentence Error Rate of 25.0% on the MEDIA test set. This means that 75% of the times the best hypothesis of CRF is correct. In turn this implies that the reranking model must not rerank 75% of times and rerank the other 25% of times, someway contrasting the evidence provided by the baseline model score. In contrast, using our *WRR* strategy, we can tune the reranking model to maximize reranking effect and recover from reranking errors applying *WRR*. As shown in tables 3 and 4, we consistently improve CRF baseline as well as reranking baseline *CRF+RR*, especially applying both SIM and WRR (*CRF+*RR_{WRR}). Comparing our results with those reported in (Hahn et al., 2010), we can see that our model reaches, and even im-

MEDIA Test set Model	OER[%]	correct found/present
CRF	9.5	2359/2657
CRF+RR	9.5	2375/2657
CRF+RR_{SIM}	7.5	2381/2758
CRF+RR_{WRR}	7.5	2444/2758

Table 5: Analysis over 10-best hypotheses for CRF baseline and the reranking models showing the effect of hypotheses selection

MEDIA Text Input **Model Pair**	**DEV** **Attr+Val**	**TEST** **Attr+Val**
CRF vs. CRF+RR	0.2235	0.4075
CRF vs. CRF+RR_{SIM}	0.0299	0.065
CRF vs. CRF+RR_{WRR}	0.0044	1.9998E-4
CRF+RR vs. CRF+RR_{SIM}	0.002	5.9994E-4
CRF+RR vs. CRF+RR_{WRR}	4.9995E-4	9.999E-5
CRF+RR_{SIM} vs. CRF+RR_{WRR}	0.1355	0.0031

Table 6: Significance tests on results of models described in this work. The significance test is based on computationally-intensive randomizations as described in (Yeh and Church, 2000).

proves in some cases, state-of-the-art performance. This is particularly meaningful since best results reported in (Hahn et al., 2010) are obtained combining 6 different SLU models.

In table 5 we report some statistics to show the effect of SIM on the 10-best hypotheses list. It is particularly interesting to see that when hypotheses selection is applied, oracle error rate (OER) drops of 2% points from an already accurate OER of 9.5%. This is reflected also by the number of oracles present in the 10-best list without applying and applying SIM. We pass from 2657 without SIM to 2758 applying our hypotheses selection metric.

Finally, in table 6 we report statistical significance tests over the models described in this work. We used the significance test described in (Yeh and Church, 2000), it is based on computationally-intensive randomizations of data and tests the null hypothesis, i.e. the lower the score, the higher the statistical significance of results difference. Scores in table 5 reflect results given in terms of CER. We can see that when the difference between results is small, this is not statistically significant, when the score is above 0.05, the difference between the two corresponding models is not significant. We can thus conclude that the reranking model we propose, using hypotheses selection and reranking errors recover, significantly improves baseline CRF model and "traditional" reranking models.

6 Discussion

Although the new ideas proposed in this paper are effective and interesting, an important issue is their applicability to other tasks and domains. In this respect, it is sufficient to note that our ideas comes from the multi-stage nature of the task and of the proposed reranking framework. SLU is performed in two intertwined steps, since attribute values are extracted from syntactic chunks annotated with concept in the first step. This allows to use the model for the second step to validate the output of the first step, and vice versa, which is the principle of our hypotheses selection metric. There are many other tasks, in NLP and in other domains, that can be modeled with multiple steps and thus the same idea of *"validation"* of the output of one step with the other's model output can be applied. An example is syntactic parsing, where in most cases parsing is performed upon POS tagging output.

7 Conclusions

In this paper we propose two improvements for reranking models to be integrated in a reranking framework for Spoken Language Understanding. The reranking model is based on a CRF baseline model and Support Vector Machines with the Partial Tree Kernel for the reraning model. The two improvements we propose are: i) hypotheses selection criteria, used before applying reranking to select better hypotheses amongst those generated by CRF. ii) a strategy to recover from reranking errors called Wrong Rerank Rejection.

We presented a full set of comparative results showing the viability of our approach. We can reach performances of state-of-the-art models, improving them in some cases, especially on automatic transcriptions coming from ASR (speech input).

In particular, the effectiveness of hypotheses selection is shown reporting the improvement of the Oracle Error Rate on the 10-best hypotheses list.

Acknowledgments

This work has been funded by OSEO under the Quaero program.

References

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, pages 142–147, Morristown, NJ, USA. Association for Computational Linguistics.

Ellen M. Voorhees. 2001. The trec question answering track. *Nat. Lang. Eng.*, 7:361–378, December.

X. Carreras and Lluis Marquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling.

R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur. 2008. Spoken language understanding: A survey. *IEEE Signal Processing Magazine*, 25:50–58.

Sylvain Galliano, Guillaume Gravier, and Maura Chaubard. 2009. The ester 2 evaluation campaign for the rich transcription of french radio broadcasts. In *Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech)*, Brighton, U.K.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, page 363370, Ann Arbor, MI.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistic (CL)*, 31(1):25–70.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282–289, Williamstown, MA, USA, June.

Brigitte Krenn and Christer Samuelsson. 1997. The linguist's guide to statistics - don't panic.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics, July.

Stefan Hahn, Patrick Lehnen, Georg Heigold, and Hermann Ney. 2009. Optimizing crfs for slu tasks in various languages using modified training criteria. In *Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech)*, Brighton, U.K.

Stefan Riezler and Alexander Vasserman. 2010. Incremental feature selection and l1 regularization for relaxed maximum-entropy modeling.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society B*, 67:301–320.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *19th Conference on Uncertainty in Artificial Intelligence*.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Mark Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

Olivier Galibert, Ludovic Quintard, Sophie Rosset, Pierre Zweigenbaum, Claire Ndellec, Sophie Aubin, Laurent Gillard, Jean-Pierre Raysz, Delphine Pois, Xavier Tannier, Louise Delger, and Dominique Laurent. 2010. Named and specific entity detection in varied data: The quro named entity baseline evaluation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Olivier Galibert. 2009. *Approches et méthodologies pour la réponse automatique à des questions adaptées un cadre interactif en domaine ouvert*. Ph.D. thesis, Universit Paris Sud, Orsay.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. 2004. The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. *Proceedings of LREC 2004*, pages 837–840.

Hélène Bonneau-Maynard, Christelle Ayache, F. Bechet, A Denis, A Kuhn, Fabrice Lefèvre, D. Mostefa, M. Qugnard, S. Rosset, and J. Servan, S. Vilaneau. 2006. Results of the french evalda-media evaluation campaign for literal understanding. In *LREC*, pages 2054–2059, Genoa, Italy, May.

Christian Raymond, Frdric Bchet, Renato De Mori, and Graldine Damnati. 2006. On the use of finite state transducers for semantic interpretation. *Speech Communication*, 48(3-4):288–304, March-April.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech)*, pages 1605–1608, Antwerp, Belgium, August.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley and Sons.

T. J. Sejnowski and C. S. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

Stefan Hahn, Marco Dinarelli, Christian Raymond, Fabrice Lefèvre, Patrick Lehen, Renato De Mori, Alessandro Moschitti, Hermann Ney, and Giuseppe Riccardi. 2010. Comparing stochastic approaches to spoken language understanding in multiple languages. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 99.

Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2009b. Re-ranking models based on small training data for spoken language understanding. In *Conference of Empirical Methods for Natural Language Processing*, pages 11–18, Singapore, August.

Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2010. Hypotheses Selection for Reranking Semantic Annotation. In *IEEE Workshop of Spoken Language Technology (SLT)*, Berkeley, USA.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of ECML 2006*, pages 318–329, Berlin, Germany.

M. Collins and N. Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete structures, and the voted perceptron. In *Proceedings of the Association for Computational Linguistics*, pages 263–270.

Libin Shen, Anoop Sarkar, and Aravind K. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Proceedings of EMNLP'06*.

Herbrich, Ralf and Graepel, Thore and Obermayer, Klaus. 2000. Large Margin Rank Boundaries for Ordinal Regression. In *Advances in Large Margin Classifiers*.

Libin Shen, and Aravind K. Joshi. 2003. An SVM Based Voting Algorithm with Application to Parse Reranking. In *Proceedings of CoNLL 2003*.

Libin Shen, Anoop Sarkar, and Franz J. Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*, pages 177–184.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

Stefan Hahn, Patrick Lehnen, and Hermann Ney. 2008a. System combination for spoken language understanding. In *Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech)*, pages 236–239, Brisbane, Australia.

J. G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (ROVER). In *Proceedings 1997 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 347–352, Santa Barbara, CA, December.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *ICML*, pages 175–182.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press.

Rush, Alexander M. and Sontag, David and Collins, Michael and Jaakkola, Tommi. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Empirical Methods for Natural Language Processing (EMNLP)*. Cambridge, Massachusetts, USA.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*, Prague, Czech Republic.

Alexander Yeh and Kelmeth Church. 2000. More accurate tests for the statistical significance of result differences.

A Fast Re-scoring Strategy to Capture Long-Distance Dependencies

Anoop Deoras
HLT-COE and CLSP
Johns Hopkins University
Baltimore MD 21218, USA
adeoras@jhu.edu,

Tomáš Mikolov
Brno University of Technology
Speech@FIT
Czech Republic
imikolov@fit.vutbr.cz,

Kenneth Church
HLT-COE and CLSP
Johns Hopkins University
Baltimore MD 21218, USA
kenneth.church@jhu.edu

Abstract

A re-scoring strategy is proposed that makes it feasible to capture more long-distance dependencies in the natural language. Two pass strategies have become popular in a number of recognition tasks such as ASR (automatic speech recognition), MT (machine translation) and OCR (optical character recognition). The first pass typically applies a weak language model (n-grams) to a lattice and the second pass applies a stronger language model to N best lists. The stronger language model is intended to capture more long-distance dependencies. The proposed method uses RNN-LM (recurrent neural network language model), which is a long span LM, to re-score word lattices in the second pass. A hill climbing method (iterative decoding) is proposed to search over *islands of confusability* in the word lattice. An evaluation based on Broadcast News shows speedups of 20 over basic N best re-scoring, and word error rate reduction of 8% (relative) on a highly competitive setup.

1 Introduction

Statistical Language Models (LMs) have received considerable attention in the past few decades. They have proved to be an essential component in many statistical recognition systems such as ASR (automatic speech recognition), MT (machine translation) and OCR (optical character recognition). The task of a language model is to assign probability to any word sequence possible in the language. The probability of the word sequence $W \equiv w_1, \ldots, w_m \equiv w_1^m$ is typically factored using the chain rule:

$$P(w_1^m) = \prod_{i=1}^{m} P(w_i|w_1^{i-1}) \quad (1)$$

In modern statistical recognition systems, an LM tends to be restricted to simple n-gram models, where the distribution of the predicted word depends on the previous $(n-1)$ words i.e. $P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-n+1}^{i-1})$.

Noam Chomsky argued that n-grams cannot learn long-distance dependencies that span over more than n words (Chomsky, 1957, pp.13). While that might seem obvious in retrospect, there was a lot of excitement at the time over the Shannon-McMillan-Breiman Theorem (Shannon, 1948) which was interpreted to say that, in the limit, under just a couple of minor caveats and a little bit of not-very-important fine print, n-gram statistics are sufficient to capture all the information in a string (such as an English sentence). Chomsky realized that while that may be true in the limit, n-grams are far from the most parsimonious representation of many linguistic facts. In a practical system, we will have to truncate n-grams at some (small) fixed n (such as trigrams or perhaps 5-grams). Truncated n-gram systems can capture many agreement facts, but not all.[1]

By long-distance dependencies, we mean facts like agreement and collocations that can span over many words. With increasing order of n-gram models we can, in theory, capture more regularities in the

[1]The discussion in this paragraph is taken as-is from an article (to appear) by Church (2012).

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1116–1127,
Edinburgh, Scotland, UK, July 27–31, 2011.

language. In addition, if we can move to more general models then we could hope to capture more, as well. However, due to data sparsity, it is hard to estimate a robust n-gram distribution for large values of n (say, $n > 10$) using the conventional Maximum Likelihood techniques, unless a more robust technique is employed for modeling which generalizes well on unseen events. Some of these well known long span / complex language models which have shown to perform very well on many speech tasks include: structured language model (Chelba and Jelinek, 2000; Roark, 2001; Wang and Harper, 2002; Filimonov and Harper, 2009), latent semantic analysis language model (Bellegarda, 2000), topic mixture language models (Iyer and Ostendorf, 1999), whole sentence exponential language models (Rosenfeld, 1997; Rosenfeld et al., 2001), feedforward neural networks (Bengio et al., 2001), recurrent neural network language models (Mikolov et al., 2010), among many others.

Although better modeling techniques can now capture longer dependencies in a language, their incorporation in decoders of speech recognition or machine translation systems becomes computationally challenging. Due to the prohibitive increase in the search space of sentence hypotheses (or longer length word sub sequences), it becomes challenging to use a long span language model in the first pass decoding. A word graph (word lattices for speech recognition systems and hypergraphs for machine translation systems), encoding exponential number of hypotheses is hence outputted at the first pass output on which a sophisticated and complex language model is deployed for re-scoring. However, sometimes even re-scoring of this refined search space can be computationally expensive due to explosion of state space.

Previously, we showed in (Deoras et al., 2011) how to tackle the problem of incorporating long span information during decoding in speech recognition systems by variationaly approximating (Bishop, 2006, pp. 462) the long span language model by a tractable substitute such that this substitute model comes closest to the long span model (closest in terms of Kullback Leibler Divergence (Cover and J.A.Thomas, 1991, pp. 20)). The tractable substitute was then used directly in the first pass speech recognition systems. In this paper we propose an approach that keeps the model intact but approximates the search space instead (which can become intractable to handle especially under a long span model), thus enabling the use of full blown model for re-scoring.With this approach, we can achieve full lattice re-scoring with a complex model, at a cost more than 20 times less than of a naive brute force approach that is commonly used today.

The rest of the paper is organized as follows: We discuss a particular form of long span language model in Sec. 2. In Sec. 3 we discuss two standard re-scoring techniques and then describe and demonstrate our proposed technique in Sec. 4. We present experimental results in Sec. 5 followed by conclusions and some remarks in Sec. 6.

2 Recurrent Neural Networks (RNN)

There is a long history of using neural networks to model sequences. Elman (1990) used recurrent neural network for modeling sentences of words generated by an artificial grammar. Work on statistical language modeling of real natural language data, together with an empirical comparison of performance to standard techniques was done by Bengio et al. (2001). His work has been followed by Schwenk (2007), who has shown that neural network language models actually work very well in the state-of-the-art speech recognition systems. Recurrent Neural Network based Language Models (RNN-LMs) (Mikolov et al., 2010) improved the ability of the original model to capture patterns in the language without using any additional features (such as part of speech, morphology etc) i.e. other than lexical ones. The RNN-LM was shown to have superior performance than the original feedforward neural network (Mikolov et al., 2011b). Recently, we also showed that this model outperforms many other advanced language modeling techniques (Mikolov et al., 2011a). We hence decided to work with this model. This model uses whole history to make predictions, thus it lies outside the family of n-gram models. Power of the model comes at a considerable computational cost. Due to the requirement of unlimited history, many optimization tricks for rescoring with feedforward-based NNLMs as presented by Schwenk (2007) cannot be applied during rescoring with RNN LM. Thus, this model is a good candidate

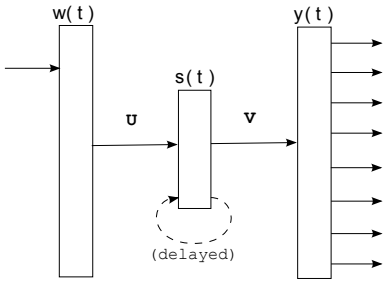


Figure 1: Schematic Representation of Recurrent Neural Network Language Model. The network has an input layer $\mathbf{w}$, a hidden layer $\mathbf{s}$ and an output layer $\mathbf{y}$. Matrices U and V represent synapses.

to show effectiveness and importance of our work.

The basic RNNLM is shown in Fig. 1. The model has an input layer $\mathbf{w}(t)$ that encodes previous word using 1 of N coding (thus, the size of the input layer is equal to the size of the vocabulary, and only the neuron that corresponds to the previous word in a sequence is set to 1). The hidden layer $\mathbf{s}(t)$ has additional recurrent connections that are delayed by one time step. After the network is trained, the output layer $\mathbf{y}(t)$ represents probability distribution for the current word, given the previous word and the state of the hidden layer from the previous time step.

The training is performed by 'backpropagation-through-time' algorithm that is commonly used for training recurrent neural networks (Rumelhart et al., 1986). More details about training, setting initial parameters, choosing size of the hidden layer etc. are presented in (Mikolov et al., 2010). Additional extensions that allow this model to be trained on large corpora are presented in (Mikolov et al., 2011b).

3 Standard Approaches for Rescoring

3.1 Word Lattice Rescoring

A word lattice, $\mathcal{L}$, obtained at the output of the first pass decoding, encodes exponential number (exponential in the number of states (nodes) present in the lattice) of hypotheses in a very compact data structure. It is a directed acyclic graph $G = (\mathcal{V}, \mathcal{E}, n_s, N_e)$, where $\mathcal{V}$ and $\mathcal{E}$ denote set of vertices (nodes / states) and edges (arcs / links), respectively. n_s and N_e denote the unique start state and set of end states.

A path, π, in a lattice is an element of $\mathcal{E}^*$ with consecutive transitions. We will denote the origin / previous state of this path by $p[\pi]$ and destination / next state of this path by $n[\pi]$. A path, π is called a *complete path* if $p[\pi] = n_s$ and $n[\pi] \in N_e$. A path, π, is called a *partial path* if $p[\pi] = n_s$ but $n[\pi]$ may or may not belong to N_e. A path, π, is called a *trailing path* if $p[\pi]$ may or may not be equal to n_s and $n[\pi] \in N_e$. We will also denote the time stamp at the start of the path by $T_s[\pi]$ and the time stamp at the end of the path by $T_e[\pi]$. Since there are nodes attached to the start and end of any path, we will denote the time stamp at any node $u \in \mathcal{V}$ by $T[u]$. Associated with every path, π, is also a word sequence $W[\pi] \in \mathcal{W}^*$, where $\mathcal{W}$ is the vocabulary used during speech recognition. For the sake of simplicity, we will distinguish word sequence of length 1 from the word sequences of length greater than 1 by using lower and upper casing i.e. $w[\cdot]$ and $W[\cdot]$ respectively.

The acoustic likelihood of the path $\pi \in \mathcal{E}^*$ is then given as:

$$A[\pi] = \prod_{j=1}^{|\pi|} P(\mathbf{a}_j | w[\pi_j])$$

where $\forall j \in \{1, 2, \ldots, |\pi|\} \quad \pi_j \in \mathcal{E}$, $\pi = \odot_{j=1}^{|\pi|} \pi_j$ and $P(\mathbf{a}_j | w[\pi_j])$ is the acoustic likelihood of the acoustic substring $\mathbf{a}_j$, spanning between $T_s[\pi_j]$ and $T_e[\pi_j]$, conditioned on the word $w[\pi_j]$ associated with the edge π_j.[2] Similarly, the language model score of the path π is given as:

$$L[\pi] = \prod_{j=1}^{|\pi|} P(w[\pi_j] | w[\pi_{j-1}], \ldots, w[\pi_{j-m+1}])$$

where $P(w[\pi_j] | w[\pi_{j-1}], \ldots, w[\pi_{j-m+1}])$ is the m-th order Markov approximation for estimating the probability of a word given the context upto that point. The speech recognizer, which uses m-th order Markov LM for first pass recognition, imposes a constraint on the word lattice such that at each state there exists an unambiguous context of consecutive $m-1$ words.

A first pass output is then a path π^* having Maximum a Posterior (MAP) probability.[3] Thus π^* is

[2] We will use $\odot$ symbol to denote concatenation of paths or word strings.

[3] Note that asterisk symbol here connotes that the path is *op-*

obtained as:

$$\pi^* = \underset{\substack{\pi: p[\pi]=n_s \\ n[\pi]\in N_e}}{\arg\max}\, A[\pi]^\gamma L[\pi],$$

where γ is the scaling parameter needed to balance the dynamic variability between the distributions of acoustic and language model (Ogawa et al., 1998). Efficient algorithms such as single source shortest path (Mohri et al., 2000) can be used for finding out the MAP path.

Under a new n-gram Language Model, rescoring involves replacing the existing language model scores of all paths π. If we denote the new language model by L_{new} and correspondingly the score of the path π by $L_{new}[\pi]$, then it is simply obtained as:

$$L_{new}[\pi] = \prod_{j=1}^{|\pi|} P(w[\pi_j]|w[\pi_{j-1}], \ldots, w[\pi_{j-n+1}])$$

where $P(w[\pi_j]|w[\pi_{j-1}], \ldots, w[\pi_{j-n+1}])$ is the n-th order Markov approximation for estimating the probability of a word given the unambiguous context of $n-1$ words under the new rescoring LM. If the Markov rescoring n-gram LM needs a bigger context for the task of prediction (i.e. $n > m$, where $m-1$ is the size of the unambiguous context maintained at every state of the word lattice), then each state of the lattice has to be split until an unambiguous context of length as large as that required by the new re-scoring language model is not maintained. The best path, π^* is then obtained as:

$$\pi^* = \underset{\substack{\pi: p[\pi]=n_s \\ n[\pi]\in N_e}}{\arg\max}\, A[\pi]^\eta L_{new}[\pi],$$

where η acts as the new scaling parameter which may or may not be equal to the old scaling parameter γ.

It should be noted that if the rescoring LM needs a context of the entire past in order to predict the next word, then the lattice has to be expanded by splitting the states many more times. This usually blows up the search space even for a reasonably small number of state splitting iterations. When the task is to do rescoring under a long span LM, such as RNN-LM, then *exact* lattice re-scoring option is not feasible. In order to tackle this problem, a suboptimal approach via N best list rescoring is utilized. The details of this method are presented next.

3.2 N best List Rescoring

N best list re-scoring is a popular way to capture some long-distance dependencies, though the method can be slow and it can be biased toward the weaker language model that was used in the first pass.

Given a word lattice, $\mathcal{L}$, top N paths $\{\pi_1, \ldots, \pi_N\}$ are extracted such that their joint likelihood under the baseline acoustic and language models are in descending order i.e. that:

$$A[\pi_1]^\gamma L[\pi_1] \geq A[\pi_2]^\gamma L[\pi_2] \geq \ldots \geq A[\pi_N]^\gamma L[\pi_N]$$

Efficient algorithms exist for extracting N best paths from word lattices (Chow and Schwartz, 1989; Mohri and Riley, 2002). If a new language model, L_{new}, is provided, which now need not be restricted to finite state machine family, then that can be deployed to get the score of the entire path π. If we denote the new LM scores by $L_{new}[\cdot]$, then under N best list paradigm, optimal path $\tilde{\pi}$ is found out such that:

$$\tilde{\pi} = \underset{\pi\in\{\pi_1,\ldots,\pi_N\}}{\arg\max}\, A[\pi]^\eta L_{new}[\pi], \qquad (2)$$

where η acts as the new scaling parameter which may or may not be equal to γ. If $N \ll |\mathcal{L}|$ (where $|\mathcal{L}|$ is the total number of complete paths in word lattice, which are exponentially many), then the path obtained using (2) is not guaranteed to be optimal (under the rescoring model). The short list of hypotheses so used for re-scoring would yield suboptimal output if the best path π^* (according to the new model) is not present among the top N candidates extracted from the lattice. This search space is thus said to be *biased* towards a weaker model mainly because the N best lists are representative of the model generating them. To illustrate the idea, we demonstrate below a simple analysis on a relatively easy task of speech transcription on WSJ data.[4] In this setup, the recognizer made use of a bi-

timal under some model. This should not be confused with the Kleene stars appearing as superscripts for $\mathcal{E}$ and $\mathcal{W}$, which serve the purpose of regular expressions implying 0 or many occurances of the element of $\mathcal{E}$ and $\mathcal{V}$ respectively.

[4] Full details about the setup can be found in (Deoras et al., 2010)

gram LM to produce lattices and hence N best lists. Each hypothesis in this set got a rank with the top most and highest scoring hypothesis getting a rank of 1, while the bottom most hypothesis getting a rank of N. We then re-scored these hypotheses with a better language model (either with a higher order Markov LM i.e. a trigram LM (*tg*) or the log linear combination of n-gram models and syntactic models (*n-gram+syntactic*) and re-ranked the hypotheses to obtain their new ranks. We then used Spearman's rank correlation factor, ρ, which takes values in $[-1, +1]$, with -1 meaning that the two ranked lists are negatively correlated (one list is in a reverse order with respect to the other list) and $+1$ meaning that the two ranked lists are positively correlated (the two lists are exactly the same). Spearman's rank correlation factor is given as:

$$\rho = 1 - \frac{6 \sum_{n=1}^{N} d_n^2}{N(N^2 - 1)}, \quad (3)$$

where d_n is the difference between the old and new rank of the n^{th} entry (in our case, difference between $n (\in \{1, 2, \ldots, N\})$ and the new rank which the n^{th} hypothesis got under the rescoring model).

Table 1 shows how the correlation factor drops dramatically when a better and a complementary LM is used for re-scoring, suggesting that the N best lists are heavily biased towards the starting models. Huge re-rankings suggests there is an opportunity to improve and also a need to explore more hypotheses, i.e. beyond N best lists.

Model	(ρ)	**WER (%)**
bg	1.00	18.2%
tg	0.41	17.4%
n-gram+syntactic	0.33	15.8%

Table 1: Spearman Rank Correlation on the N best list extracted from a bi-gram language model (*bg*) and re-scored with relatively better language models including, trigram LM (*tg*), and the log linear combination of n-gram models, and syntactic models (*n-gram+syntactic*). With a bigger and a better LM, the WER decreases at the expense of huge re-rankings of N best lists, only suggesting the fact that N best lists generated under a weaker model, are not reflective enough of a relatively better model.

In the next section, we propose an algorithm which keeps the representation of search space as simple as that of N best list, but does not restrict itself to top N best paths alone and hence does not get *biased* towards the starting weaker model.

4 Proposed Approach for Rescoring

A high level idea of our proposed approach is to identify *islands of confusability* in the word lattice and replace the problem of global search over word lattice by series of local search problems over these islands in an iterative manner. The motivation behind this strategy is the observation that the recognizer produces bursts of errors such that they have a temporal scope. The recognizer output (sentence hypotheses) when aligned together typically shows a pattern of confusions both at the word level and at the phrase level. Regions where there are singleton words competing with one another (reminiscent of a confusion bin of a Confusion Network (CN) (Mangu, 2000)), choice of 1 word edit distance works well for the formation of local neighborhood. Regions where there are phrases competing with other phrases, choice of variable length neighborhood works well. Previously, Richardson et al. (1995) demonstrated a hill climbing framework by exploring 1 word edit distance neighborhood, while in our own previous work (Deoras and Jelinek, 2009), we demonstrated working of iterative decoding algorithm, a hill climbing framework, for CNs, in which the neighborhood was formed by all words competing with each other in any given time slot, as defined by a confusion bin.

In this work, we propose a technique which generalizes very well on word lattices and overcomes the limitations posed by a CN or by the limited nature of local neighborhood. The size of the neighborhood in our approach is a *variable* factor which depends upon the confusability in any particular region of the word lattice. Thus the local neighborhood are in some sense a function of the confusability present in the lattice rather than some predetermined factor. Below we describe the process, virtue of which, we can cut the lattice to form many self contained smaller sized sub lattices. Once these sub lattices are formed, we follow a similar hill climbing procedure as proposed in our previous work (Deoras and Jelinek, 2009).

4.1 Islands of Confusability

We will continue to follow the notation introduced in section 3.1. Before we define the procedure for cutting the lattice into many small self contained lattices, we will define some more terms necessary for the ease of understandability of the algorithm.[5] For any node $v \in \mathcal{V}$, we define forward probability, $\alpha(v)$, as the probability of any partial path $\pi \in \mathcal{E}^*$, s.t. $p[\pi] = n_s, n[\pi] = v$ and it is given as:

$$\alpha(v) = \sum_{\substack{\pi \in \mathcal{E}^* \\ s.t. p[\pi]=n_s, n[\pi]=v}} A[\pi]^\gamma L[\pi] \qquad (4)$$

Similarly, for any node $v \in \mathcal{V}$, we define the backward probability, $\beta(v)$, as the probability of any trailing path $\pi \in \mathcal{E}^*$, s.t. $p[\pi] = v, n[\pi] \in N_e$ and it is given as:

$$\beta(v) = \sum_{\substack{\pi \in \mathcal{E}^* \\ s.t. p[\pi]=v, n[\pi] \in N_e}} A[\pi]^\gamma L[\pi] \qquad (5)$$

If we define the sum of joint likelihood under the baseline acoustic and language models of all paths in the lattice by Z, then it can simply be obtained as: $Z = \sum_{u \in N_e} \alpha(u) = \beta(n_s)$

In order to cut the lattice, we want to identify sets of nodes, $S_1, S_2, \ldots, S_{|\mathcal{S}|}$ such that for any set $S_i \in \mathcal{S}$ following conditions are satisfied:

1. For any two nodes $u, v \in S_i$ we have that: $T[u] = T[v]$. We will define this common time stamp of the nodes in the set by $T[S_i]$.
2. $\nexists\, \pi \in \mathcal{E}$ such that $T_s[\pi] < T[S_i] < T_e[\pi]$.

The first property can be easily checked by first pushing states into a linked list associated with each time marker (this can be done by iterating over all the states of the graph) then iterating over the unique time markers and retrieving back the nodes associated with it.The second property can be checked by first iterating over the unique time markers and for each of the marker, iterating over the arcs and terminating the loop as soon as some arc is found out violating property 2 for the specific time marker. Thus the time complexity for checking property 1 is $O(|\mathcal{V}|)$ and that for property 2 is $O(|\mathcal{T}| \times |\mathcal{E}|)$, where $|\mathcal{T}|$ is the total number of unique time markers. Usually $|\mathcal{T}| \ll |\mathcal{E}|$ and hence the time complexity for checking property 2 is almost linear in the number of edges. Thus effectively, the time complexity for cutting the lattice is $O(|\mathcal{V}| + |\mathcal{E}|)$.

Having formed such sets, we can now cut the lattice at time stamps associated with these sets i.e. that: $T[S_1], \ldots, T[S_{|\mathcal{S}|}]$. It can be easily seen that the number of sub lattices, C, will be equal to $|\mathcal{S}| - 1$.We will identify these sub lattices as $\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C$. At this point, we have not formed self contained lattices yet by simply cutting the parent lattice at the cut points.

Once we cut the lattice at these cut points, we implicitly introduce many new starting nodes and ending nodes for any sub lattice. We will refer to these nodes as exposed starting nodes and exposed ending nodes. Thus for some j^{th} sub lattice, $\mathcal{L}_j$, there will be as many new exposed starting nodes as there are nodes in the set S_j and as many exposed ending nodes as there are nodes in the set S_{j+1}. In order to make these sub lattices consistent with the definition of a word lattice (see Sec. 3.1), we unify all the exposed starting nodes and exposed ending nodes. To unify the exposed starting nodes, we introduce as many *new* edges as there are nodes in the set S_j such that they have a common starting node, $n_s[\mathcal{L}_j]$, (newly created) and distinct ending nodes present in S_j. To unify the exposed ending nodes of $\mathcal{L}_j$, we introduce as many *new* edges as there are nodes in the set S_{j+1} such that they have distinct starting nodes present in S_{j+1} and a common ending node $n_e[\mathcal{L}_j]$ (newly created). From the totality of these new edges and nodes along with the ones already present in $\mathcal{L}_j$ forms an induced directed acyclic subgraph $G[\mathcal{L}_j] = (\mathcal{V}[\mathcal{L}_j], \mathcal{E}[\mathcal{L}_j], n_s[\mathcal{L}_j], n_e[\mathcal{L}_j])$.

For any path $\pi \in \mathcal{E}[\mathcal{L}_j]$ such that $p[\pi] = n_s[\mathcal{L}_j]$ and $n[\pi] \in S_j$, we assign the value of $\alpha(n[\pi])$ to denote the joint likelihood $A[\pi]^\gamma L[\pi]$ and assign $epsilon$ for word associated with these edges i.e. $w[\pi]$. We assign $T[S_j] - \delta T$ to denote $T_s[\pi]$ and $T[S_j]$ to denote $T_e[\pi]$. Similarly, for any path $\pi \in \mathcal{E}[\mathcal{L}_j]$ such that $p[\pi] \in S_{j+1}$ and $n[\pi] = n_e[\mathcal{L}_j]$,

[5] Goel and Byrne (2000) previously demonstrated the lattice segmentation procedure to solve the intractable problem of MBR decoding. The cutting procedure in our work is different from theirs in the sense that we rely on time information for collating competing phrases, while they do not.

we assign the value of $\beta(p[\pi])$[6] to denote the joint likelihood $A[\pi]^{\gamma}L[\pi]$ and assign $epsilon$ for word associated with these edges i.e. $w[\pi]$. We assign $T[S_{j+1}]$ to denote $T_s[\pi]$ and $T[S_{j+1}] + \delta T$ to denote $T_e[\pi]$. This completes the process and we obtain self contained lattices, which if need be, can be independently decoded and/or analyzed.

4.2 Iterative Decoding on Word Lattices

Once we have formed the self contained lattices, $\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C$, where C is the total number of sub lattices formed, then the idea is to divide the re-scoring problem into many small re-scoring problems carried over the sub lattices one at a time by fixing single best paths from all the remaining sub lattices.

The inputs to the algorithm are the sub lattices (produced by cutting the parent lattice generated under some Markov n-gram LM) and a new rescoring LM, which now need not be restricted to finite state machine family. The output of the algorithm is a word string, $\mathbf{W}^*$, such that it is the concatenation of final decoded word strings from each sub lattice. Thus if we denote the final decoded path (under some decoding scheme, which will become apparent next) in the j^{th} sub lattice by π_j^* and the concatenation symbol by '$\cdot$', then $\mathbf{W}^* = W[\pi_1^*] \cdot W[\pi_2^*] \cdot \ldots \cdot W[\pi_C^*] = \odot_{j=1}^{C} W[\pi_j^*]$.

Algorithm 1 Iterative Decoding on word lattices.

Require: $\{\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C\}$, L_{new}
  PrevHyp $\leftarrow$ null
  CurrentHyp $\leftarrow \odot_{j=1}^{C} W[\hat{\pi}_j]$
  while PrevHyp $\neq$ CurrentHyp **do**
    for $i \leftarrow 1 \ldots C$ **do**

$$\hat{\pi}_i \leftarrow \underset{\substack{\pi_i \in \mathcal{E}_i^*: \\ p[\pi_i]=n_s[\mathcal{L}_i] \\ n[\pi_i]=n_e[\mathcal{L}_i]}}{\operatorname{argmax}} \Big(L_{new}[\hat{\pi_1} \cdot \ldots \cdot \pi_i \cdot \ldots \cdot \hat{\pi_k}] \times A[\pi_i]^{\eta} \prod_{\substack{j=1 \\ j \neq i}}^{k} A[\hat{\pi_j}]^{\eta} \Big)$$

    end for
    PrevHyp $\leftarrow$ CurrentHyp
    CurrentHyp $\leftarrow \odot_{j=1}^{C} W[\hat{\pi}_j]$
  end while
  $\forall j \in \{1, 2, \ldots, C\} \quad \pi_j^* \leftarrow \hat{\pi}_j$

[6]The values of $\alpha(\cdot)$ and $\beta(\cdot)$ are computed under parent lattice structure.

The algorithm is initialized by setting **PrevHypo** to null and **CurrHypo** to the concatenation of 1-best output from each sub lattice. During the initialization step, each sub lattice is analyzed independent of any other sub lattice and under the baseline acoustic scores and baseline n-gram LM scores, 1-*best* path is found out. Thus if we define the best path under baseline model in some j^{th} sub-lattice by $\hat{\pi}_j$, **CurrHypo** is then initialized to: $W[\hat{\pi}_1] \cdot W[\hat{\pi}_2] \cdot \ldots \cdot W[\hat{\pi}_C]$. The algorithm then runs as long as CurrHypo is not equal to PrevHypo. In each iteration, the algorithm sequentially re-scores each sub-lattice by keeping the surrounding context fixed. Once all the sub lattices are re-scored, that constitutes one iteration. At the end of each iteration, CurrHypo is set to the concatenation of 1 best paths from each sub lattice while PrevHypo is set to the old value of CurrHypo. Thus if we are analyzing some i^{th} sub-lattice in some iteration, then 1-best paths from all but this sub-lattice is kept fixed and a *new* 1-*best* path under the re-scoring LM is found out. It is not hard to see that the likelihood of the output under the new re-scoring model is guaranteed to increase monotonically after every decoding step.

Since the cutting of parent lattices produce many small lattices with considerably lesser number of nodes, in practice, an exhaustive search for the 1-best hypothesis can be carried out via N best list. Algorithm 1 outlines the steps for iterative decoding on word lattices.

4.3 Entropy Pruning

In this section, we will discuss a speed up technique based on entropy of the lattice. Entropy of a lattice reflects the confidence of the recognizer in recognizing the acoustics. Based on the observation that if the N best list / lattice generated under some model has a very low entropy, then the Spearman's rank correlation factor, ρ (Eqn. 3), tends to be higher even when the N best lists / lattice is re-ranked with a bigger and a better model. A low entropy under the baseline model only reflects the confidence of the recognizer in recognizing the acoustic. Table 2 shows the rank correlation values between two sets of N best lists. Both sets are produced by a bigram LM (bg). The entropy of N best lists in the first set is 0.05 nats or less. The N best lists in the second set have an entropy greater than 0.05 nats.

Both these sets are re-ranked with bigger and better models (see Table 1 for model definitions). We can see from Table 2 that the rank correlation values tend to be higher (indicating little re-rankings) when the entropy of the N best list, under the baseline model, is lower. Similarly, the rank-correlation values tend to be lower (indicating more re-rankings) whenever the entropy of the N best list is higher. Note that these entropy values are computed with respect to the starting model (in this case, bigram LM). Of course, if the starting LM is much weaker than the rescoring model, then the entropy values need not be reflective of the difficulty of the overall task. This observation then suggests that it is safe to rescore only those N best lists whose entropy under the starting model is higher than some threshold.

Rescoring Model	$\rho_{(H \leq 0.05)}$	$\rho_{(H > 0.05)}$
bg	1.00	1.00
tg	0.58	0.38
n-gram+syntactic	0.54	0.31

Table 2: Spearman Rank Correlation on the N best list extracted from a bi-gram language model (*bg*) and rescored with relatively better language models (see Table 1 for model definitions). Entropy under the baseline model correlates well with the rank correlation factor, suggesting that exhaustive search need not be necessary for utterances yielding lower entropy.

While computation of entropy for N best list is tractable, for a word lattice, the computation of entropy is intractable if one were to enumerate all the hypotheses. Even if we were able to enumerate all hypotheses, this method tends to be slower. Using efficient semiring techniques introduced by Li and Eisner (2009) or using posterior probabilities on the edges leading to end states, we can compute the entropy of a lattice in one single forward pass using dynamic programming. It should, however, be noted that, for dynamic programming technique to work, only n-gram LMs can be used. One has to resort to approximate entropy computation via N best list, if entropy under long span LM is desired.

4.3.1 Speed Up for Iterative Decoding

Our speed up technique is simple. Once we have formed self contained sub lattices, we want to prune all but the top few best complete paths (obtained under baseline / starting model) of those sub lattices whose entropy is below some threshold. Thus, believing in the original model's confidence, we want to focus only on those sub lattices which the recognizer found difficult to decode in the first pass. All other part of the parent lattice will be not be analyzed. The thresholds for pruning is very application and corpus specific and needs to be tuned on some held out data.

5 Experiments and Results

We performed recognition on the Broadcast News (BN) dev04f, rt03 and rt04 task using the state-of-the-art acoustic models trained on the English Broadcast News (BN) corpus (430 hours of audio) provided to us by IBM (Chen et al., 2009). IBM also provided us its state-of-the-art speech recognizer, Attila (Soltau et al., 2010) and two Kneser-Ney smoothed backoff n-gram LMs containing 4.7M n-grams ($n \leq 4$) and 54M n-grams ($n \leq 4$), both trained on 400M word tokens. We will refer to them as KN:BN-Small and KN:BN-Big respectively. We refer readers to (Chen et al., 2009) for more details about the recognizer and corpora used for training the models.

We trained two RNN based language models - the first one, denoted further as RNN-limited, was trained on a subset of the training data (58M tokens). It used 400 neurons in the hidden layer. The second model, denoted as RNN-all, was trained on all of the training data (400M tokens), but due to the computational complexity issues, we had to restrict its hidden layer size to 320 neurons.

We followed IBM's multi-pass decoding recipe using KN:BN-Small in the first pass followed by either N best list re-scoring or word lattice re-scoring using bigger and better models.[7] For the purpose of re-scoring, we combined all the relevant statistical models in one unified log linear framework reminiscent of work by Beyerlein (1998). We, however, trained the model weights by optimizing expected WER rather than 1-*best* loss as described in (Deoras et al., 2010). Training was done on N best lists of size 2K. We will refer to the log linear com-

[7]The choice of the order and size of LM to be used in the first pass decoding was determined by taking into consideration the capabilities of the decoder.

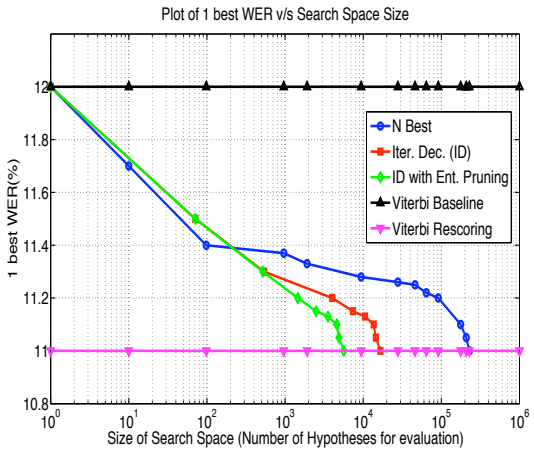


Figure 2: Plot of WER (y axis) on rt03+dev04f set versus the size of the search space (x axis). The baseline WER obtained using KN:BN-Small is 12% which then drops to 11% when KN:BN-Big is used for re-scoring. N best list search method obtains the same reduction in WER by evaluating as many as 228K sentence hypotheses on an average. The proposed method obtains the same reduction by evaluating 14 times smaller search space. The search effort reduces further to 40 times if entropy based pruning is employed during re-scoring.

bination of KN:BN-Big and RNN-limited by KN-RNN-lim; KN:BN-Big and RNN-all by KN-RNN-all and KN:BN-Big, RNN-limited and RNN-all by KN-RNN-lim-all.

We used two sets for decoding: rt03+dev04f set was used as a development set while rt04 was used as a blind set for the purpose of evaluating the performance of long span RNN models using the proposed approach. We made use of OpenFst C++ libraries (Allauzen et al., 2007) for manipulating lattice graphs and generating N best lists. Due to the presence of hesitation tokens in reference transcripts and the need to access the silence/pause tokens for penalizing short sentences, we treated these tokens as regular words before extracting sentence hypotheses. This, and poorly segmented nature of the test corpora, led to huge enumeration of sentence hypotheses.

5.1 n-gram LM for re-scoring

In this setup, we used KN:BN-Small as the baseline starting LM which yielded the WER of 12% on rt03+dev04f set. Using KN:BN-Big as the re-scoring LM, the WER dropped to 11%. Since the re-scoring LM belonged to the n-gram family, it was possible to compute the optimal word string by re-scoring the whole lattice (see Sec. 3.1). We now compare the performance of N best list approach (Sec. 3.2) with our proposed approach (Sec. 4). N best list achieved the best possible reduction by evaluating as many as 228K sentence hypotheses on an average. As against that, our proposed approach achieved the same performance by evaluating 16.6K sentence hypotheses, thus reducing the search efforts by **13.75** times. By carrying out entropy pruning (see Sec. 4.3) on sub lattices, our proposed approach required as little as 5.6K sentence hypotheses evaluations to obtain the same optimal performance, reducing the search effort by as much as **40.46** times. For the purpose of this experiment, entropy based pruning was carried out when the entropy of the sub lattice was below 5 nats. Table 3 compares the two search methods for this setup and Fig. 2 shows a plot of WER versus the size of the search space (in terms of number of sentence hypotheses evaluated by an n-gram language model).

On rt04, the KN:BN-Small LM gave a WER of 14.1% which then dropped to 13.1% after re-scoring with KN:BN-Big. Since the re-scoring model was an n-gram LM, it was possible to obtain the optimal performance via lattice update technique (see Sec. 3.1). We then carried out the re-scoring of the word lattices under KN:BN-Big using our proposed technique and found it to give the same performance yielding the WER of 13.1%.

5.2 Long Span LM for re-scoring

In this setup, we used the strongest n-gram LM as our baseline. We thus used KN:BN-Big as the baseline LM which yielded the WER of 11% on rt03+dev04f. We then used KN-RNN-lim-all for re-scoring. Due to long span nature of the re-scoring LM, it was not possible to obtain the optimal WER performance. Hence we have compared the performance of our proposed method with N best list approach. N best list achieved the lowest possible WER after evaluating as many as 33.8K sentence hypotheses on an average. As against that, our proposed approach in conjunction with entropy pruning obtained the same performance by evaluating just 1.6K sentence hypotheses, thus reducing the search by a factor of **21**. Fig 3 shows a plot of WER versus

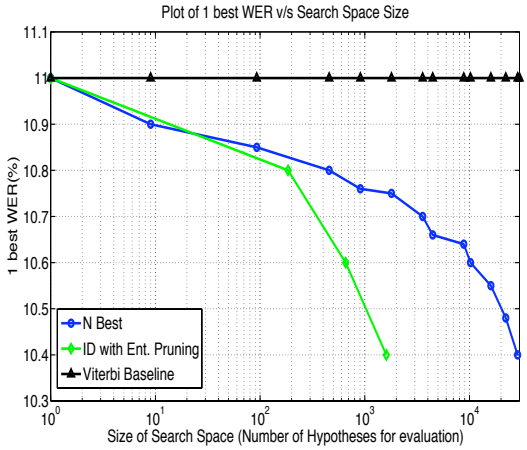


Figure 3: Plot of WER (y axis) on rt03+dev04f set versus the size of the search space (x axis). The baseline WER obtained using KN:BN-Big is 11% which then drops to 10.4% when KN-RNN-lim-all is used for re-scoring. N best list search method obtains this reduction in WER by evaluating as many as 33.8K sentence hypotheses on an average, while the proposed method (with entropy pruning) obtains the same reduction by evaluating 21 times smaller search space.

the size of the search space (in terms of number of sentence hypotheses evaluated by a long span language model).

In-spite of starting off with a very strong n-gram LM, the N best lists so extracted were still not representative enough of the long span rescoring models. Had we started off with KN:BN-Small, the N best list re-scoring method would have had no chance of finding the optimal hypothesis in reasonable size of hypotheses search space. Table 4 compares the two search methods for this setup when many other long span LMs were also used for re-scoring.

On rt04, the KN:BN-Big LM gave a WER of 13.1% which then dropped to **12.15**% after re-scoring with KN-RNN-lim-all using our proposed technique.[8] Since the re-scoring model was not an n-gram LM, it was not possible to obtain the optimal performance but we could enumerate huge N best list to approximate this value. Our proposed method is much faster than huge N best lists and no worse in terms of WER. As far as we know, the result obtained on these sets is the best performance ever reported on the Broadcast News corpus for speech recognition.

[8]The WER obtained using KN-RNN-lim and KN-RNN-all were 12.5% and 12.3% respectively.

Models	WER	NBest	ID	Saving
KN:BN-Small	12.0	-	-	-
KN:BN-Big	11.0	228K	5.6K	40

Table 3: The starting LM is a weak n-gram LM (KN:BN-Small) and the re-scoring LM is a much stronger but n-gram LM (KN:BN-Big). The baseline WER in this case is 12% and the optimal performance by the re-scoring LM is 11.0%. The proposed method outperforms N best list approach, in terms of search efforts, obtaining optimal WER.

Models	WER	NBest	ID	Saving
KN:BN-Big	11.0	-	-	-
KN-RNN-lim	10.5	42K	1.1K	38
KN-RNN-all	10.5	26K	1.3K	20
KN-RNN-lim-all	10.4	34K	1.6K	21

Table 4: The starting LM is a strong n-gram LM (KN:BN-Big) and the re-scoring model is a long span LM (KN-RNN-*). The baseline WER is 11.0%. Due to long span nature of the LM, optimal WER could not be estimated. The proposed method outperfoms N best list approach on every re-scoring task.

6 Conclusion

We proposed and demonstrated a new re-scoring technique for general word graph structures such as word lattices. We showed its efficacy by demonstrating huge reductions in the search effort to obtain a new state-of-the-art performance on a very competitive speech task of Broadcast news. As part of the future work, we plan to extend this technique for hypergraphs and lattices in re-scoring MT outputs with complex and long span language models.

Acknowledgement

This work was partly funded by Human Language Technology, Center of Excellence and by Technology Agency of the Czech Republic grant No. TA01011328, and Grant Agency of Czech Republic project No. 102/08/0707. We would also like to acknowledge the contribution of Frederick Jelinek towards this work. He would be a co-author if he were available and willing to give his consent.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer.

J. R. Bellegarda. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of IEEE*, 88(8):1279–1296.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A Neural Probabilistic Language Model. In *Proceedings of Advances in Neural Information Processing Systems*.

Peter Beyerlein. 1998. Discriminative Model Combination. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Ciprian Chelba and Frederick Jelinek. 2000. Structured Language Modeling. *Computer Speech and Language*, 14(4):283–332.

S. F. Chen, L. Mangu, B. Ramabhadran, R. Sarikaya, and A. Sethy. 2009. Scaling shrinkage-based language models. In *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 299–304.

Noam Chomsky. 1957. *Syntactic Structures*. The Hague: Mouton.

Yen-Lu Chow and Richard Schwartz. 1989. The N-Best algorithm: an efficient procedure for finding top N sentence hypotheses. In *Proceedings of the workshop on Speech and Natural Language*, HLT '89, pages 199–202, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kenneth Church. 2012. A Pendulum Swung Too Far. *Linguistic Issues in Language Technology - LiLT*. to appear.

T.M. Cover and J.A.Thomas. 1991. *Elements of Information Theory*. John Wiley and Sons, Inc. N.Y.

Anoop Deoras and Frederick Jelinek. 2009. Iterative Decoding: A Novel Re-Scoring Framework for Confusion Networks. In *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 282 –286.

Anoop Deoras, Denis Filimonov, Mary Harper, and Fred Jelinek. 2010. Model Combination for Speech Recognition using Empirical Bayes Risk Minimization. In *Proc. of IEEE Workshop on Spoken Language Technology (SLT)*.

Anoop Deoras, Tomáš Mikolov, Stefan Kombrink, Martin Karafiát, and Sanjeev Khudanpur. 2011. Variational Approximation of Long-Span Language Models for LVCSR. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Jeffery Elman. 1990. Finding Structure in Time. In *Cognitive Science*, volume 14, pages 179–211.

Denis Filimonov and Mary Harper. 2009. A Joint Language Model with Fine-grain Syntactic Tags. In *Proc. of 2009 Conference on Empirical Methods in Natural Language Processing*.

V. Goel and W. Byrne. 2000. Minimum Bayes Risk Automatic Speech Recognition. *Computer, Speech and Language*.

Rukmini Iyer and Mari Ostendorf. 1999. Modeling Long Distance Dependence in Language: Topic Mixtures Versus Dynamic Cache Models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 40–51, Singapore, August.

Lidia Luminita Mangu. 2000. *Finding consensus in speech recognition*. Ph.D. thesis, The Johns Hopkins University. Adviser-Brill, Eric.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Proc. of the ICSLP-Interspeech*.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan "Honza" Černocký. 2011a. Empirical Evaluation and Combination of Advanced Language Modeling Techniques. In *Proc. of Interspeech*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur. 2011b. Extensions of Recurrent Neural Network Language Model. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Mehryar Mohri and Michael Riley. 2002. An Efficient Algorithm for the N-Best-Strings Problem. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

M. Mohri, F.C.N. Pereira, and M. Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science, 231:17-32*.

A. Ogawa, K. Takeda, and F. Itakura. 1998. Balancing Acoustic and Linguistic Probabilities. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

F. Richardson, M. Ostendorf, and J.R. Rohlicek. 1995. Lattice-based search strategies for large vocabulary speech recognition. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Roni Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. 2001. Whole-Sentence Exponential Language Models: a Vehicle for Linguistic-Statistical Integration. *Computer Speech and Language*, 15(1).

Roni Rosenfeld. 1997. A Whole Sentence Maximum Entropy Language Model. In *Proc. of IEEE workshop on Automatic Speech Recognition and Understanding (ASRU)*, Santa Barbara, California, December.

D.E. Rumelhart, G. E. Hinton, and R.J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

C. E. Shannon. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 623–656.

H. Soltau, G. Saon, and B. Kingsbury. 2010. The IBM Attila speech recognition toolkit. In *Proc. of IEEE Workshop on Spoken Language Technology (SLT)*.

Wen Wang and Mary Harper. 2002. The SuperARV language model: investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Efficient Subsampling for Training Complex Language Models

Puyang Xu
puyangxu@jhu.edu

Asela Gunawardana[#]
aselag@microsoft.com

Sanjeev Khudanpur
khudanpur@jhu.edu

Department of Electrical and Computer Engineering
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218, USA

[#]Microsoft Research
Redmond, WA 98052, USA

Abstract

We propose an efficient way to train maximum entropy language models (MELM) and neural network language models (NNLM). The advantage of the proposed method comes from a more robust and efficient subsampling technique. The original multi-class language modeling problem is transformed into a set of binary problems where each binary classifier predicts whether or not a particular word will occur. We show that the binarized model is as powerful as the standard model and allows us to aggressively subsample negative training examples without sacrificing predictive performance. Empirical results show that we can train MELM and NNLM at $1\% \sim 5\%$ of the standard complexity with no loss in performance.

1 Introduction

Language models (LM) assign probabilities to sequences of words. They are widely used in many natural language processing applications. The probability of a sequence can be modeled as a product of local probabilities, as shown in (1), where w_i is the i^{th} word, and h_i is the word history preceding w_i.

$$P(w_1, w_2, ..., w_l) = \prod_{i=1}^{l} P(w_i|h_i) \qquad (1)$$

Therefore the task of language modeling reduces to estimating a set of conditional distributions $\{P(w|h)\}$. The n-gram LM is a dominant way to parametrize $P(w|h)$, where it is assumed that w only depends on the previous $n-1$ words. More complex models have also been proposed–MELM (Rosenfeld, 1996) and NNLM (Bengio et al., 2003) are two examples.

Modeling $P(w|h)$ can be seen as a multi-class classification problem. Given the history, we have to choose a word in the vocabulary, which can easily be a few hundred thousand words in size. For complex models such as MELM and NNLM, this poses a computational challenge for learning, because the resulting objective functions are expensive to normalize. In contrast, n-gram LMs do not suffer from this computational challenge. In the web era, language modelers have access to virtually unlimited amounts of data, while the computing power available to process this data is limited. Therefore, despite the demonstrated effectiveness of complex LMs, the n-gram is still the predominant approach for most real world applications.

Subsampling is a simple solution to get around the constraint of computing resources. For the purpose of language modeling, it amounts to taking only part of the text corpus to train the LM. For complex models such as NNLM, it has been shown that subsampling can speed up training greatly, at the cost of some degradation in predictive performance (Schwenk, 2007), allowing for trade-off between computational cost and LM quality.

Our contribution is a novel way to train complex LMs such as MELM and NNLM which allows much more aggressive subsampling without incurring as high a cost in predictive performance. The key to our approach is reducing the multi-class LM problem into a set of *binary* problems. Instead of training a V-class classifier, where V is the size of

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1128–1136,
Edinburgh, Scotland, UK, July 27–31, 2011.

the vocabulary, we train V binary classifiers, each one of which performs a *one-against-all* classification. The V trained binary probabilities are then renormalized to obtain a valid distribution over the V words. Subsampling here can be done in the *negative* examples. Since the majority of training examples are negative for each of the binary classifiers, we can achieve substantial computational saving by only keeping subsets of them. We will show that the binarized LM is as powerful as its multi-class counterpart, while being able to sustain much more aggressive subsampling. For certain types of LMs such as MELM, there are more benefits–the binarization leads to a set of completely independent classifiers to train, which allows easy parallelization and significantly lowers the memory requirement.

Similar one-against-all approaches are often used in the machine learning community, especially by SVM (support vector machine) practitioners to solve multi-class problems (Rifkin and Klautau, 2004; Allwein et al., 2000). The goal of this paper is to show that a similar technique can also be used for language modeling and that it enables us to subsample data much more efficiently. We show that the proposed approach is useful when the dominant modeling constraint is computing power as opposed to training data.

The rest of the paper is organized as follows. In section 2, we describe our binarization and subsampling techniques for language models with MELM and NNLM as two specific examples. Experimental results are presented in Section 3, followed by discussion in Section 4.

2 Approximating Language Models with Binary Classifiers

Suppose we have an LM that can be written in the form

$$P(w|h) = \frac{\exp a_w(h;\theta)}{\sum_{w'} \exp a_{w'}(h;\theta)}, \quad (2)$$

where $a_w(h;\theta)$ is a parametrized history representation for word w.

Given a training corpus of word history pairs with empirical distribution $\tilde{P}(h,w)$, the regularized log likelihood of the training set can be written as

$$\mathcal{L} = \sum_h \tilde{P}(h) \sum_w \tilde{P}(w|h) \log P(w|h) - r(\theta), \quad (3)$$

where $r(\theta)$ is the regularizing function over the parameters.

Assuming that $r(\theta)$ can be written as a sum over per-word regularizers, namely $r(\theta) = \sum_w r_w(\theta)$, we can take the gradient of the log likelihood w.r.t θ to show that the regularized MLE for the LM satisfies

$$\sum_h \tilde{P}(h) \sum_w P(w|h) \nabla_\theta a_w(h;\theta)$$
$$= \sum_{h,w} \tilde{P}(w,h) \nabla_\theta a_w(h;\theta) - \sum_w \nabla_\theta r_w(\theta). \quad (4)$$

For each word w, we can define a binary classifier that predicts whether the next word is w by

$$P_b(w|h) = \frac{\exp a_w(h;\theta)}{1 + \exp a_w(h;\theta)}. \quad (5)$$

The regularized training set log likelihood for all the binary classifiers is given by

$$\mathcal{L}_b = \sum_w \sum_h \tilde{P}(h) \Big[\tilde{P}(w|h) \log P_b(w|h)$$
$$+ \tilde{P}(\bar{w}|h) \log P_b(\bar{w}|h) \Big] - \sum_w r_w(\theta), \quad (6)$$

where $P_b(\bar{w}|h) = 1 - P_b(w|h)$ is the probability of w not occurring. Here we assume the same structure of the regularizer $r(\theta)$.

The regularized MLE for the binary classifiers satisfies

$$\sum_h \tilde{P}(h) \sum_w P_b(w|h) \nabla_\theta a_w(h;\theta)$$
$$= \sum_{h,w} \tilde{P}(w,h) \nabla_\theta a_w(h;\theta) - \sum_w \nabla_\theta r_w(\theta). \quad (7)$$

Notice the right hand sides of (4) and (7) are the same. Thus, taking $P'(w|h) = P_b(w|h)$ from ML trained binary classifiers gives an LM that meets the MLE constraints for language models. Therefore, if $\sum_w P_b(w|h) = 1$, ML training for the language model is equivalent to ML training of the binary classifiers and using the probabilities given by the classifiers as our LM probabilities.

Note that in practice, the probabilities given by the binary classifiers are not guaranteed to sum up to one. For tasks such as measuring perplexity,

these probabilities have to be normalized explicitly. Our hope is that for large enough data sets and rich enough history representation $a_w(h;\theta)$, we will get $\sum_w P_b(w|h) \approx 1$ so that renormalizing the classifiers to get

$$P'(w|h) = \frac{P_b(w|h)}{\sum_{w' \in V} P_b(w'|h)} \tag{8}$$

will not change the MLE constraint too much.

2.1 Stratified Sampling

We note that iterative estimation of the LM shown in (2) in general requires enumerating over the T training cases in the training set and computing the denominator of (2) for each case at a cost of $O(V)$. Thus, each iteration of training takes $O(VT)$ in general. The complexity of estimating each of the V binary classifiers is $O(T)$ per iteration, also giving $O(VT)$ per iteration in total.

However, as mentioned earlier, we are able to maximally subsample negative examples for each classifier. Thus the classifier for w is trained using the $C(w)$ positive examples and a proportion α of the $T - C(w)$ negative examples. The total number of training examples for all V classifiers is then $(1-\alpha)T + \alpha VT$. For large V, we choose $\alpha >> \frac{1}{1+V}$ so that this is approximately αVT. Thus, our complexity for estimating all V classifiers is $O(\alpha VT)$.

The resulting training set for each binary classifier is a stratified sample (Neyman, 1934), and our estimate needs to be calibrated to account for this. Since the training set subsamples negative examples by α, the resulting classifier will have a likelihood ratio

$$\frac{P_b(w|h)}{1 - P_b(w|h)} = \exp a_w(h;\theta) \tag{9}$$

that is overestimated by a factor of $\frac{1}{\alpha}$. This can be corrected by simply adding $\log \alpha$ to the bias (unigram) weight of the classifier.

2.2 Maximum Entropy LM

MELM is an effective alternative to the standard n-gram LM. It provides a flexible framework to incorporate different knowledge sources in the form of feature constraints. Specifically, MELM takes the form of (2), for word w following history h, we have the following probability definition,

$$P(w|h) = \frac{\exp \sum_i \theta_i f_i(h,w)}{\sum_{w' \in V} \exp \sum_i \theta_i f_i(h,w')}. \tag{10}$$

f_i is the i^{th} feature function defined over the word-history pair, θ_i is the feature weight associated with f_i. By defining general features, we have a natural framework to go beyond n-grams and capture more complex dependencies that exist in language. Previous research has shown the benefit of including various kinds of syntactic and semantic information into the LM (Khudanpur and Wu, 2000). However, despite providing a promising avenue for language modeling, MELM are computationally expensive to estimate. The bottleneck lies in the denominator of (10).

To estimate θ_is, gradient based methods can be used. The derivative of the likelihood function $\mathcal{L}$ w.r.t θ_i has a simple form, namely

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \sum_k f_i(w_k, h_k) - \sum_k \sum_{w' \in V} P(w'|h) f_i(w', h_k), \tag{11}$$

where k is the index of word-history pair in the training corpus. The first term in the derivative is the observed feature count in the training corpus, the second term is the expected feature count according to the model. In order to obtain $P(w'|h)$ in the second term, we need to compute the normalizer, which involves a very expensive summation over the entire vocabulary. As described earlier, the complexity for each iteration of training is at $O(VT)$, where T is the size of training corpus.

For feature sets that can be expressed hierarchically, for example n-gram feature set, where higher order n-grams imply lower order n-grams, Wu and Khudanpur (2000) exploit the structure of the normalizer, and precompute components that can be shared by different histories. For arbitrary feature sets, however, it may not be possible to establish the required hierarchical relations and the normalizer still needs to be computed explicitly. Goodman (2001) changes the original LM into a class-based LM, where each one of the two-step predictions only involves a much smaller summation in the normalizer. In addition, MELM estimation can be parallelized, with expected count computation done

separately for different parts of the training data and merged together at the end of each iteration. For models with massive parametrizations, this merge step can be expensive due to communication costs.

Obviously, a different way to expedite MELM training is to simply train on less data. We propose a way to do this without incurring a significant loss of modeling power, by reframing the problem in terms of binary classification. As mentioned above, we build V binary classifiers of the form in (5) to model the distribution over the V words. The binary classifiers use the same features as the MELM of (10), and are given by:

$$P_b(w|h) = \frac{\exp \sum_i \theta_i f_i(h, w)}{1 + \exp \sum_i \theta_i f_i(h, w)}. \quad (12)$$

We assume the features are partitioned over the vocabulary, so that each feature f_i has an associated w such that $f_i(h, w') = 0$ for all $w' \neq w$. Therefore, the corresponding θ_i affects only the binary classifier for w. This gives an important advantage in terms of parallelization–we have a set of binary classifiers with no feature sharing, and can be trained separately on different machines. The parallelized computations are completely independent and do not require the tedious communication between machines. Memory-wise, since the computations are independent, each word trainer only have to store features that are associated with the word, so the memory requirement for each individual worker is significantly reduced.

2.3 Neural Network LM

Neural Network Language Models (NNLM) have gained a lot of interest since their introduction (Bengio et al., 2003). While in standard language modeling, words are treated as discrete symbols, NNLM map them into a continuous space and learn their representations automatically. It is often believed that NNLM can generalize better to sequences that are not seen in the training data. However, despite having been shown to outperform standard n-gram LM (Schwenk, 2007), NNLM are computationally expensive to train.

Figure 1 shows the standard feed-forward NNLM architecture. Starting from the left part of the figure, each word of the $n-1$ words history is mapped to a

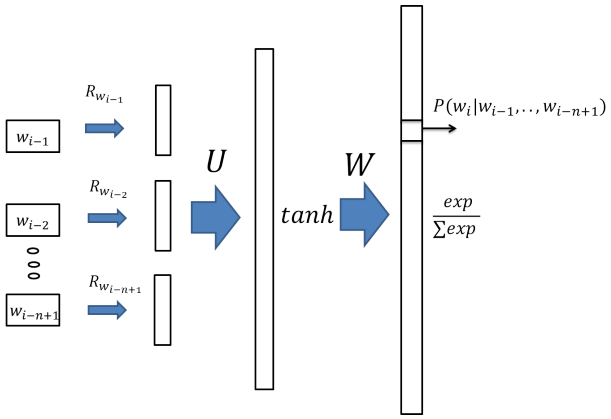


Figure 1: Feed-forward NNLM

continuous vector and concatenated. Through a nonlinear hidden layer, the neural network constructs a multinomial distribution at the output layer. Denoting the concatenated d-dimensional word representations $\mathbf{r}$, we have the following probability definition:

$$P(w_i = k|w_{i-1}, ..., w_{i-n+1}) = \frac{e^{a_k}}{\sum_m e^{a_m}}, \quad (13)$$

$$a_k = b_k + \sum_{l=1}^{h} W_{kl} \tanh(c_l + \sum_{j=1}^{(n-1)d} U_{lj} r_j), \quad (14)$$

where h denotes the hidden layer size, $\mathbf{b}$ and $\mathbf{c}$ are the bias vectors for the output nodes and hidden nodes respectively. Note that NNLM also has the form of (2).

Stochastic gradient descent is often used to maximize the training data likelihood under such a model. The gradient can be computed using the back-propagation method. To analyze the complexity, computing an n-gram conditional probability requires approximately

$$O((n-1)dh + h + Vh + V) \quad (15)$$

operations, where V is the size of the vocabulary. The four terms in the complexity correspond to computing the hidden layer, applying the nonlinearity, computing the output layer and normalization, respectively. The error propagation stage can be analyzed similarly and takes about the same number of operations. For typical values as used in our experiments, namely $n = 3$, $d = 50$, $h = 200$,

$V = 10000$, the majority of the complexity per iteration comes from the term hV. For large scale tasks, it may be impractical to train an NNLM.

A lot of previous research has focused on speeding up NNLM training. It usually aims at removing the computational dependency on V. Schwenk (2007) used a short list of frequent words such that a large number of out-of-list words are taken care of by a back-off LM. To reduce the gradient computation introduced by the normalizer, Bengio and Senecal (2008) proposed a different kind of importance sampling technique. A recent work (Mikolov et al., 2011) applied Goodman's class MELM trick (2001) to NNLM, in order to avoid the gigantic normalization. A similar technique has been introduced even earlier which took the idea of factorizing output layer to the extreme (Morin, 2005) by replacing the V-way prediction by a tree-style hierarchical prediction. The authors show a theoretical complexity reduction from $O(V)$ to $(\log V)$, but the technique requires a careful clustering which may not be easily attainable in practice.

Subsampling has also been proposed to accelerate NNLM training (Schwenk, 2007). The idea is to select random subsets of the training data in each epoch of stochastic gradient descent. After some epochs, it is very likely that all of the training examples have been seen by the model. We will show that our binary classifier representation leads to a more robust and promising subsampling strategy.

As with MELM, we notice that the parameters of (14) can be interpreted as also defining a set of V per-word binary classifiers

$$P_b(w_i = k|w_{i-1}, ..., w_{i-n+1}) = \frac{e^{a_k}}{1 + e^{a_k}}, \quad (16)$$

but with a common hidden layer representation. As in MELM, we will train the classifiers, and renormalize them to obtain an NNLM over the V words.

In order to train the classifiers, we need to compute all V output nodes and propagate the errors back. Since the hidden layer is shared, the classifiers are not independent, and the computations can not be easily parallelized to multiple machines. However, subsampling can be done differently for each classifier. Each training instance serves as a positive example for one classifier and as a negative example for only a fraction α of the others. The rest of the nodes are not computed and do not produce error signal for the hidden representation. We calibrate the classifiers after subsampled training as described above for MELM.

It is straightforward to show that the dominating term Vh in the complexity is reduced to αVh. We want to point out that compared with MELM, subsampling the negatives here does not always reduce the complexity proportionally. In cases where the vocabulary is very small, as shown in (15), computing the hidden layer can no longer be ignored. Nonetheless, real world applications such as speech recognition, usually involves a vocabulary of considerable size, therefore, subsampling in the binary setting can still achieve substantial speedup for NNLM.

3 Experimental Results

3.1 MELM

We evaluate the proposed technique on two datasets of different sizes. Our first dataset is obtained from Penn Treebank. Section 00-20 are used for training(972K tokens), section 21-22 are the validation set(77K), section 23-24(86K) are the test set. The vocabulary size of the experiment is $10,000$. This is one of the standard setups on which many researchers have reported perplexity results on (Mikolov et al., 2011).

The binary MELM is trained using stochastic gradient descent, no explicit regularization is performed (Zhang, 2004). The learning rate starts at 0.1 and is halved every time the perplexity on the validation set stops decreasing. It usually takes around 20 iterations before no significant improvement can be obtained on the validation set. The training stops at that time.

We compare perplexity with both the standard interpolated Kneser-Ney trigram model and the standard MELM. The MELM is L^2 regularized and estimated using a variant of generalized iterative scaling, the regularizer is tuned on the validation data. To demonstrate the effectiveness of our subsampling approach, we compare the subsampled versions of the binary MELM and the standard MELM. In order to obtain valid perplexities, the binary LMs are first renormalized explicitly according to equation (8) for each test history.

Model	PPL
KN Trigram	153.0
Standard MELM, Feat-I	154.2
Binary MELM, Feat-I	153.7
Standard MELM, Feat-II	140.2
Binary MELM, Feat-II	141.1

Table 1: Binary MELM vs. Standard MELM

We consider two kinds of feature sets: Feat-I contains only n-gram features, namely unigram, bigram and trigram features, with no count cutoff, the total number of features is $0.9M$. Feat-II is augmented with skip-1 bigrams and skip-1 trigrams (Goodman, 2001), as well as word trigger features as described in (Rosenfeld, 1996). The total number of features in this set is $1.9M$. Note that the speedup trick described in (Wu and Khudanpur, 2000) can be used for feat-I, but not feat-II.

Table 1 shows the perplexity results when no subsampling is performed. With only n-gram features, the binary MELM is able to match both standard MELM and the Kneser-Ney model. We can also see that by adding features that are known to be able to improve the standard MELM, we can get the same improvement in the binary setting.

Figure 2 shows the comparisons of the two types of MELM when the training data are subsampled. The standard MELM with n-gram features suffers drastically as we sample more aggressively. In contrast, the binary n-gram MELM(Feat-I) does not appear to be hurt by aggressive subsampling, even when 99% of the negative examples are discarded. The robustness also holds for Feat-II where more complicated features are added into the model. This suggests a very efficient way of training MELM– with only 1% of the computational cost, we are able to train an LM as powerful as the standard MELM.

We further test our approach on a second dataset which comes from Wall Street Journal corpus. It contains 26M training tokens and a test set of 22K tokens. We also have a held-out validation set to tune parameters. This set of experiments is intended to demonstrate that the binary subsampling technique is useful on a large text corpus where training a standard MELM is not practical, and gives a better LM than the commonly used Kneser-Ney baseline.

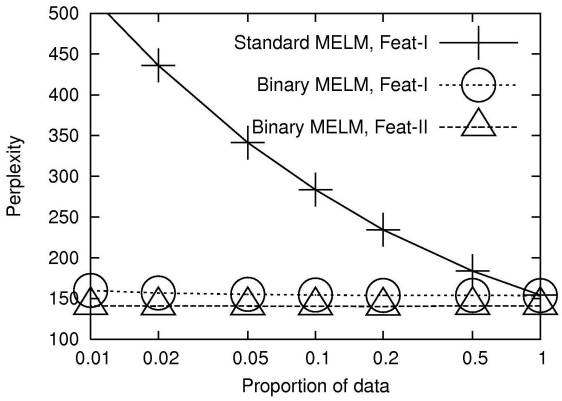


Figure 2: Subsampled Binary MELM vs. Subsampled Standard MELM

Model	PPL
KN Trigram	117.7
Standard MELM, Trigram	116.5
Binary MELM, Feat-III, 10%	110.2
Binary MELM, Feat-III, 5%	110.8
Binary MELM, Feat-III, 2%	112.1
Binary MELM, Feat-III, 1%	112.4

Table 2: Binary Subsampled MELM on WSJ

The binary MELM is trained in the same way as described in the previous experiment. Besides unigram, bigram and trigram features, we also added skip-1 bigrams and skip-1 trigrams, this gives us 7.5M features in total. We call this set of features feat-III. We were unable to train a standard MELM with feat-III or a binary MELM without subsampling because of the computational cost. However, with our binary subsampling technique, as shown in Table 2, we are able to benefit from skip n-gram features with only 5% of the standard MELM complexity. Also the performance does not degrade much as we discard more negative examples.

To show that such improvement in perplexity translates into gains in practical applications, we conducted a set of speech recognition experiments. The task is on Wall Street Journal, the LMs are trained on 37M tokens and are used to rescore the n-best list generated by the first pass recognizer with a trigram LM. The details of the experimental setup can be found in (Xu et al., 2009). Our baseline LM is an interpolated Kneser-Ney 4-gram model.

Note that the size of the vocabulary for the task

Model	Dev WER	Eval WER
KN 4-gram	11.8	17.2
Binary MELM, Feat-IV, 5%	11.0	16.7
Binary MELM, Feat-IV, 2%	11.2	16.7
Binary MELM, Feat-IV, 1%	11.2	16.7

Table 3: WSJ WER improvement. Binary MELM are interpolated with KN 4-gram

Model	PPL			
	100%	20%	10%	5%
Standard NNLM	154.3	239.8	297.0	360.3
Binary NNLM	-	152.7	160.0	176.2
KN trigram	153.0	-	-	-

Table 4: Binary NNLM vs. Standard NNLM. Fixed random subset.

Model	Interpolated PPL			
	100%	20%	10%	5%
Standard NNLM	132.7	145.6	148.6	150.7
Binary NNLM	-	132.1	134.2	138.0
KN trigram	153.0	-	-	-

Table 5: Binary NNLM vs. Standard NNLM. Fixed random subset. Interpolated with KN trigram.

is 20K, for the purpose of rescoring, we are only interested in the words that exist in the n-best list, therefore, for the binary MELM, we only have to train about 5300 binary classifiers. For comparison, the KN 4-gram also uses the same restricted vocabulary. The features for the binary MELM are n-gram features up to 4-grams plus skip-1 bigrams and skip-1 trigrams. The total number of features is 10M. We call this set of features Feat-IV.

Table 3 demonstrates the word error rate(WER) improvement enabled by our binary subsampling technique. Note that we can achieve 0.5% absolute WER improvement on the test set at only 1% of the standard MELM complexity. More specifically, with only 50 machines, such a reduction in complexity allows us to train a binary MELM with skip n-gram features in less than two hours, which is not possible for the standard MELM on 37M words.

Obviously, with more machines, the estimation can be even faster, it's also reasonable to expect that with more kinds of features, the improvement can be even larger. We think that the proposed technique opens the door for the utilization of the modeling framework provided by MELM at a scale that has not been possible before.

3.2 NNLM

We evaluate our binary subsampling technique on the same Penn Treebank corpus as described for the MELM experiments. Taking random subsets of the training data with the standard model is our primary baseline to compare with. The NNLM we train is a trigram LM with $tanh$ hidden units. The size of word representation and the size of hidden layer are tuned minimally on the validation set(Hidden layer size 200; Representation size 50). We adopt the same learning rate strategy as for training MELM, and the validation set is used to track perplexity performance and adjust learning rate correspondingly. All parameters are initialized randomly with mean 0 and variance 0.01. As with binary MELM, binary NNLM are explicitly renormalized to obtain valid perplexities.

In our first experiment, we keep the subsampled data fixed as we did for MELM. For the standard NNLM, it means only a subset of the data is seen by the model and it does not change through epochs; For binary NNLM, it means the subset of negative examples for each binary classifier does not change. Table 4 shows the perplexity results by NNLM itself and the interpolated results are shown in Table 5.

We can see that both models exhibit a tendency to deteriorate as we subsample more aggressively. However, the standard NNLM is clearly impacted more severely. With binary NNLM, we are able to retain all the gain after interpolation with only 20% of the negative examples.

Notice that with a fixed random subset, we are not replicating the experiments of Schwenk (Schwenk, 2007) exactly, although it is reasonable to expect both models are able to benefit from seeing different random subsets of the training data. This is verified by results in Table 6 and Table 7.

The standard NNLM benefits quite a lot going from using a fixed random subset to a variable random subset, but still demonstrates a clear tendency to deteriorate as we discard more and more data. On the constrast, the binary NNLM maintains all the performance gain with only 5% of the negative examples and still clearly outperforms its counterpart.

Model	PPL			
	100%	20%	10%	5%
Standard NNLM	154.3	157.7	172.2	186.5
Binary NNLM	-	151.7	150.1	152.1

Table 6: Binary NNLM vs. Standard NNLM. Variable random subset.

Model	Interpolate PPL			
	100%	20%	10%	5%
Standard NNLM	132.7	133.9	138.1	141.2
Binary NNLM	-	132.2	131.7	132.2

Table 7: Binary NNLM vs. Standard NNLM. Variable random subset. Interpolated with KN trigram.

4 Discussion

For the standard models, the amount of existent patterns fed into training heavily depends on the subsampling rate α. For a small α, the models will inevitably lose some training patterns given any reasonable number of epochs of training. Taking variable random subsets in each epoch can alleviate this problem to some extent, but still can not solve the fundamental problem. In the binary setting, we are able to do subsampling differently. While the complexity remains the same without subsampling, the majority of the complexity comes from processing negatives examples for each binary classifier. Therefore, we can achieve the same level of speedup as standard subsampling by only subsampling negative examples, and most importantly, it allows us to keep all the existent patterns(positive examples) in the training data. Of course, negative examples are important and even in the binary case, we benefit from including more of them, but since we have so many of them, they might not be as critical as positive examples in determining the distribution.

A similar conclusion can be drawn from Google's work on large LMs (Brants et al., 2007). Not having to properly smooth the LM, they are still able to benefit from large volumes of web text as training data. It is probably more important to have a high n-gram coverage than having a precise distribution.

The explanation here might lead us to wonder whether for the multi-class problem, subsampling the terms in the normalizer would achieve the same results. More specifically, instead of summing over all words in the vocabulary, we may choose to only consider α of them. In fact, the short-list approach in (Schwenk, 2007) and the adaptive importance sampling in (Bengio and Senecal, 2008) have exactly this intuition. However, in the multi-class setup, subsampling like this has to be very careful. We have to either have a good estimate of how much probability mass we've thrown away, as in the short-list approach, or have a good estimate of the entire normalizer, as in the importance sampling approach. It is very unlikely that an arbitrary random subsampling will not harm the model. Fortunately, in the binary case, the effect of random subsampling is much easier to analyze. We know exactly how much negative examples we've discarded, and they can be compensated easily in the end.

It is worth pointing out that the proposed technique is not restricted to MELM and NNLM. We have done experiments to binarize the class trick sometimes used for language modeling (Goodman, 2001; Mikolov et al., 2011), and it also proves to be useful. We plan to report these results in the future. More generally, for many large-scale multi-class problems, binarization and subsampling can be an effective combination to consider.

5 Conclusion

We propose efficient subsampling techniques for training large multi-class classifiers such as maximum entropy language models and neural network language models. The main idea is to replace a multi-way decision by a set of binary decisions. Since most of the training instances in the binary setting are negatives examples, we can achieve substantial speedup by subsampling only the negatives. We show by extensive experiments that this is more robust than subsampling subsets of training data for the original multi-class classifier. The proposed method can be very useful for building large language models and solving more general multi-class problems.

Acknowledgments

This work is partially supported by National Science Foundation Grant No 0963898, the DARPA GALE Program and JHU/HLTCOE.

References

Allwein, Erin, Robert Schapire, Yoram Singer and Pack Kaelbling. 2000. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1:113-141.

Bengio, Yoshua, Rejean Ducharme and Pascal Vincent 2003. A neural probabilistic language model *Journal of Machine Learning research*, 3:1137–1155.

Bengio, Yoshua and J. Senecal 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model *IEEE Transaction on Neural Network*, Apr. 2008.

Berger, Adam, Stephen A. Della Pietra and Vicent J. Della Pietra 1996. A Maximum Entropy approach to Natural Language Processing. *Computational Linguistics*, 1996, 22:39-71.

Brants, Thorsten, Ashok C. Popat, Peng Xu, Frank J. Och and Jeffrey Dean 2007. Large language models in machine translation. *In Proceedings of 2007 Conference on Empirical Methods in Natural Language Processing*, 858–867.

Goodman, Joshua 2001. Classes for Fast Maximum Entropy Training. *Proceedings of 2001 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Goodman, Joshua 2001. A bit of Progress in Language Modeling. *Computer Speech and Language*, 403-434.

Khudanpur, Sanjeev and Jun Wu 2000. Maximum Entropy Techniques for Exploiting Syntactic, Semantic and Collocational Dependencies in Language Modeling. *Computer Speech and Language*, 14(4):355-372.

Mikolov, Tomas, Stefan Kombrink, Lukas Burget, Jan "Honza" Cernocky and Sanjeev Khudanpur 2011. Extensions of recurrent neural network language model. *Proceedings of 2011 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Morin, Frederic 2005. Hierarchical probabilistic neural network language model. *AISTATS'05*, pp. 246-252.

Neyman, Jerzy 1934. On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection. *Journal of the Royal Statistical Society*, 97(4):558-625.

Rifkin, Ryan and Aldebaro Klautau 2004. In Defense of One-Vs-All Classification. *Journal of Machine Learning Research*.

Rosenfeld, Roni. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228.

Schwenk, Holger 2007. Continuous space language model. *Computer Speech and Language*, 21(3):492-518.

Wu, Jun and Sanjeev Khudanpur. 2000. Efficient training methods for maximum entropy language modeling. *Proceedings of the 6th International Conference on Spoken Language Technologies*, pp. 114–117.

Xu, Puyang, Damianos Karakos and Sanjeev Khudanpur. 2009. Self-supervised discriminative training of statistical language models. *Proceedings of 2009 IEEE Automatic Speech Recognition and Understanding Workshop*.

Zhang, Tong 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. *Proceedings of 2004 International Conference on Machine Learnings*.

Generating Aspect-oriented Multi-Document Summarization with Event-aspect model

Peng Li[1] and **Yinglin Wang**[1] and **Wei Gao**[2]and **Jing Jiang**[3]
[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong
[3] School of Information Systems, Singapore Management University
{lipeng, ylwang@sjtu.edu.cn} {wgao@se.cuhk.edu.hk} {jingjiang@smu.edu.sg}

Abstract

In this paper, we propose a novel approach to automatic generation of aspect-oriented summaries from multiple documents. We first develop an event-aspect LDA model to cluster sentences into aspects. We then use extended LexRank algorithm to rank the sentences in each cluster. We use Integer Linear Programming for sentence selection. Key features of our method include automatic grouping of semantically related sentences and sentence ranking based on extension of random walk model. Also, we implement a new sentence compression algorithm which use dependency tree instead of parser tree. We compare our method with four baseline methods. Quantitative evaluation based on Rouge metric demonstrates the effectiveness and advantages of our method.

1 Introduction

In recent years, there has been much interest in the task of multi-document summarization. In this paper, we study the task of automatically generating aspect-oriented summaries from multiple documents. The goal of aspect-oriented summarization is to present the most important content to the user in a condensed form and a well-organized structure to satisfy the user's needs. A summary should follow a readable structure and cover all the aspects users are interested in. For example, a summary about natural disasters should include aspects about what happened, when/where it happened, reasons, damages, rescue efforts, etc. and these aspects may be scattered in multiple articles written by different news agencies. Our goal is to automatically collect aspects and construct summaries from multiple documents.

Aspect-oriented summarization can be used in many scenarios. First of all, it can be used to generate Wikipedia-like summary articles, especially used to generate introduction sections that summarizes the subject of articles before the table of contents and other elaborate sections. Second, opinionated text often contains multiple viewpoints about an issue generated by different people. Summarizing these multiple opinions can help people easily digest them. Furthermore, combined with search engines and question&answering systems, we can better organize the summary content based on aspects to improve user experience.

Despite its usefulness, the problem of modeling domain specific aspects for multi-document summarization has not been well studied. The most relevant work is by (Haghighi and Vanderwende, 2009) on exploring content models for multi-document summarization. They proposed a HIERSUM model for finding the subtopics or aspects which are combined by using KL-divergence criterion for selecting relevant sentences. They introduced a general content distribution and several specific content distributions to discover the topic and aspects for a single document collection. However, the aspects may be shared not only across documents in a single collection, but also across documents in different topic-related collections. Their model is conceptually inadequate for simultaneously summarizing multiple topic-related document collections. Furthermore, their sentence selection method based on KL-divergence cannot prevent redundancy across different aspects.

In this paper, we study how to overcome these

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1137–1146,
Edinburgh, Scotland, UK, July 27–31, 2011.

limitations. We hypothesize that comparatively summarizing topics across similar collections can improve the effectiveness of aspect-oriented multi-document summarization. We propose a novel extraction-based approach which consists of four main steps listed below:

Sentence Clustering: Our goal in this step is to automatically identify the different aspects and cluster sentences into aspects (See Section 2). We substantially extend the entity-aspect model in (Li et al., 2010) for generating general sentence clusters.

Sentence Ranking: In this step, we use an extension of LexRank algorithm proposed by (Paul et al., 2010) to score representative sentences in each cluster (See Section 3).

Sentence Compression: In this step, we aim to improve the linguistic quality of the summaries by simplifying the sentence expressions. We prune sentences using grammatical relations defined on dependency trees for recognizing important clauses and removing redundant subtrees (See Section 4).

Sentence Selection: Finally, we select one compressed version of the sentences from each aspect cluster. We use Integer Linear Programming (ILP) algorithm, which optimizes a *global* objective function, for sentence selection (McDonald, 2007; Gillick and Favre, 2009; Sauper and Barzilay, 2009) (See Section 5).

We evaluate our method using TAC2010 Guided Summarization task data sets[1] (Section 6). Our evaluation shows that our method obtains better ROUGE recall score compared with four baseline methods, and it also achieve reasonably high-quality aspect clusters in terms of purity.

2 Sentence Clustering

In this step, our goal is to discover event aspects contained in a document set and cluster sentences into aspects. Here we substantially extend the entity-aspect model in Li et al. (2010) and refer to it as event-aspect model. The main difference between our event-aspect model and entity-aspect model is that we introduce an additional layer of event topics and the separation of general and specific aspects.

Our extension is based upon the following observations. For example, specific events like "Columbine Massacre" and "Malaysia Resort Abduction" can be related to the "Attack" topic. Each event consists of multiple articles written by different news agencies. Interesting aspects may include "what happened, when, where, perpetrators, reasons, who affected, damages and countermeasures," etc[2]. We compared the "Columbine Massacre" and "Malaysia Resort Abduction" data sets and found 5 different kinds of words in the text: (1) stop words that occur frequently in any document collection; (2) general content words describing "damages" or "countermeasures" aspect of attacks; (3) specific content words describing "what happened", "who affected" or "where" aspect of the concrete event; (4) background words describing the general topic of "Attack"; (5) words that are local to a single document and do not appear across different documents. Table 1 shows four sentences related to two major aspects. We found that the entity-aspect model does not have enough capacity to cluster sentences into aspects (See Section 6). So we introduce additional layer to improve the effectiveness of sentence clustering. We also found that their one aspect per sentence assumption is not very strong in this scenario. Although a sentence may belong to a single general aspect, it still contains multiple specific aspect words like second sentence in Table 1. Therefore, We assume that each sentence belongs to both a general aspect and a specific aspect.

2.1 Event-Aspect Model

Stop words can be ignored by LDA model because they can be easily identified using a standard stop word list. Suppose that for a given event topic, there are in total C specific events for which we need to simultaneously generate summaries. We can assume four kinds of unigram language models (i.e. multinomial word distributions). For each event topic, there is a background model $\phi^{\mathcal{B}}$ that generates words commonly used in all documents, and there are A^G general aspect models ϕ^{ga} ($1 \leq ga \leq A^G$), where A^G is the number of general aspects. For each specific event in a topic, there are A^S specific aspect

[1] http://www.nist.gov/tac/2010/Summarization/

[2] http://www.nist.gov/tac/2010/Summarization/Guided-Summ.2010.guidelines.html

countermeasures
Police/GA are/S close/B to/S identifying/GA someone/B responsible/GA for/S the/S attack/B .
Investigators/GA do/S not/S know/B how/S many/S suspects/SA they/S are/S looking/B for/S, but/S reported/B progress/B toward/S identifying/GA one/S of/S the/S bombers/SA .
what happened, when, where
During/S the/S morning/SA rush/D hour/D on/S July/SA 7/SA terrorists/B exploded/SA bombs/SA on/S three/D London/SA subway/D trains/SA and/S a/S double-decker/D bus/SA .
Four/D coordinated/B bombings/SA struck/B central/B London/SA on/SA July/SA 7/SA, three/D in/S subway/D cars/SA and/S one/D on/S a/S bus/SA .

Table 1: Four sentences on "COUNTERMEASURES" and "What, When, Where" aspects from the "Attack" topic. S: stop word. B: background word. GA: general aspect word. SA: specific aspect word. D: document word.

models ϕ^{sa} ($1 \leq sa \leq A^S$), where A^S is the number of specific aspects, and also there are D document models ϕ^d ($1 \leq d \leq D$), where D is the number of documents in this collection. We assume that these word distributions have a uniform Dirichlet prior with parameter β.

We introduce a level distribution σ that controls whether we choose a word from ϕ^{ga} or ϕ^{sa}. σ is sampled from $Beta(\delta_0, \delta_1)$ distribution. We also introduce an aspect distribution θ that controls how often a general or a specific aspect occurs in the collection, where θ is sampled from another Dirichlet prior with parameter α. There is also a multinomial distribution π that controls in each sentence how often we encounter a background word, a document word, or an aspect word. π has a Dirichlet prior with parameter γ.

Let S_d denote the number of sentences in document d, $N_{d,s}$ denote the number of words (after stop word removal) in sentence s of document d, and $w_{d,s,n}$ denote the n'th word in this sentence. We introduce hidden variables $z^{ga}_{d,s}$ and $z^{sa}_{d,s}$ to indicate that a sentence s of document d belongs to which general or specific aspects . We introduce hidden variables $y_{d,s,n}$ for each word to indicate whether a word is generated from the background model, the document model, or the aspect model. We also introduce hidden variables $l_{d,s,n}$ to indicate whether the n'th word in sentence s of document d is generated from the general aspect model. Figure 1 describes the process of generating the whole document collection. The plate notation of the model is shown in Figure 2. Note that the values of δ_0, δ_1, α_1, α_2, β and γ are fixed. The number of general and specific aspects A^G and A^S are also empirically set.

Given a document collection, i.e. the set of all $w_{d,s,n}$, our goal is to find the most likely assignment of $z^{ga}_{d,s}$, $z^{sa}_{d,s}$, $y_{d,s,n}$ and $l_{d,s,n}$ that maximizes distribution $p(\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{l}|\boldsymbol{w}; \alpha, \beta, \gamma, \delta)$, where $\boldsymbol{z}$, $\boldsymbol{y}$, $\boldsymbol{l}$ and $\boldsymbol{w}$ represent the set of all z, y, l and w variables, respectively. With the assignment, sentences are naturally clustered into aspects, and words are labeled as either a background word, a document word, a general aspect word or a specific aspect word.

Inference can be done with Gibbs sampling, which is commonly used in LDA models (Griffiths and Steyvers, 2004).

In our experiments, we set $\alpha_1 = 5$, $\alpha_2 = 3$, $\beta = 0.01$, $\gamma = 20$, $\delta_1 = 10$ and $\delta_2 = 10$. We run 100 burn-in iterations through all documents in a collection to stabilize the distribution of $\boldsymbol{z}$ and $\boldsymbol{y}$ before collecting samples. We take 10 samples with a gap of 10 iterations between two samples, and average over these 10 samples to get the estimation for the parameters.

After estimating all the distributions, we can find the values of each $z^{ga}_{d,s}$ and $z^{sa}_{d,s}$ that gives us sentences clustered into general and specific aspects.

3 Sentence Ranking

In this step, we want to order the clustered sentences so that the representative sentences can be ranked higher in each aspect. Inspired by Paul et al. (2010), we use an extended LexRank algorithm to obtain top ranked sentences. LexRank (Erkan and Radev, 2004) algorithm defines a random walk mod-

1. Draw $\theta_1 \sim \text{Dir}(\alpha_1), \theta_2 \sim \text{Dir}(\alpha_2), \pi \sim \text{Dir}(\gamma)$ Draw $\sigma \sim \text{Beta}(\delta_0, \delta_1)$
2. For each event topic, there is a background model $\phi^{\mathcal{B}}$, and there are general aspect ga, where $1 \leq ga \leq A^G$
   (a) draw $\phi^{\mathcal{B}} \sim \text{Dir}(\beta)$
   (b) draw $\phi^{ga} \sim \text{Dir}(\beta)$
3. For each document collection, there are specific aspect sa, where $1 \leq sa \leq A^S$
   (a) draw $\phi^{sa} \sim \text{Dir}(\beta)$
4. For each document $d = 1, \ldots, D$,
   (a) draw $\phi^d \sim \text{Dir}(\beta)$
   (b) for each sentence $s = 1, \ldots, S_d$
      i. draw $z^{ga} \sim \text{Multi}(\theta_1)$
      ii. draw $z^{sa} \sim \text{Multi}(\theta_2)$
      iii. for each word $n = 1, \ldots, N_{d,s}$
         A. draw $l_{d,s,n} \sim \text{Binomial}(\sigma)$
         B. draw $y_{d,s,n} \sim \text{Multi}(\pi)$
         C. draw $w_{d,s,n} \sim \text{Multi}(\phi^{\mathcal{B}})$ if $y_{d,s,n} = 1$, $w_{d,s,n} \sim \text{Multi}(\phi^d)$ if $y_{d,s,n} = 2$, $w_{d,s,n} \sim \text{Multi}(\phi^{z^{sa}_{d,s}})$ if $y_{d,s,n} = 3$ and $l_{d,s,n} = 1$ or $w_{d,s,n} \sim \text{Multi}(\phi^{z^{ga}_{d,s}})$ if $y_{d,s,n} = 3$ and $l_{d,s,n} = 0$

Figure 1: The document generation process.

el on top of a graph that represents sentences to be summarized as nodes and their similarities as edges. The LexRank score of a sentence gives the expected probability that a random walk will visit that sentence in the long run. A variant is called continuous LexRank improved LexRank by making use of the strength of the similarity links. The continuous LexRank score can be computed using the following formula:

$$L(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} p(u|v) L(v)$$

where $L(u)$ is the LexRank value of sentence u, N is the total number of nodes in the graph, d is a damping factor for the convergence of the method, and $p(u|v)$ is the jumping probability between sentence u and its neighboring sentence v. $p(u|v)$ is defined using content similarity function $sim(u, v)$ between two sentences:

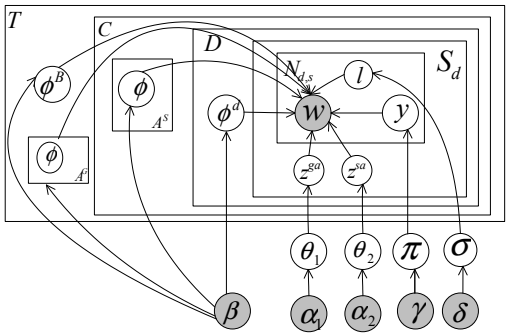


Figure 2: The event-aspect model.

$$p(u|v) = \frac{sim(u, v)}{\sum_{z \in adj[v]} sim(z, v)}$$

The major extension is to modify this jumping probability so as to favor visiting representative sentences. More specifically, we scale $sim(u, v)$ by the likelihood that the two sentences represent the same general aspect ga or specific aspect sa:

$$sim'(u, v) = sim(u, v)[\sum_{ga=1}^{A^G} P(ga|u)P(ga|v) + \sum_{sa=1}^{A^S} P(sa|u)P(sa|v)]$$

where the value $P(ga|u)$ and $P(sa|u)$ can be computed by our event-aspect model. We define $sim(u, v)$ as the $tf * idf$ weighted cosine similarity between two sentences.

We found that sentence ranking is better conducted before the compression because the pre-compressed sentences are more informative and the similarity function in LexRank can be better off with the complete information.

4 Sentence Compression

It has been shown that sentence compression can improve linguistic quality of summaries (Zajic et al., 2007; Gillick et al., 2010). Commonly used "Syntactic parse and trim" approach may produce poor compression results. For example, given the sentence "We have friends whose children go to Columbine, the freshman said", the procedure tries to remove the clause "the freshman said" from the parse tree by using the "SBAR" label to locate the

clause, and will result in "whose children go to Columbine", which is not adequate. Furthermore, some important temporal modifier, numeric modifier and clausal complement need to be retained because they reflect content aspects of the summary. Therefore, we propose the "dependency parse and trim" approach, which prunes sentences based on dependency tree representations, using English grammatical relations to recognize clauses and remove redundant structures. Table 2 shows two examples by removing redundant auxiliary clauses. Below is the sentence compression procedure:

1. Select possible subtree root nodes using grammatical relations, such as clausal complement, complementizer, or parataxis [3].

2. Decide which subtree root node can be the root of clause. If this root contains maximum number of child nodes and the collection of all child edges include object or auxiliary relations, it is selected as the root node.

3. Remove redundant modifiers such as adverbials, relative clause modifiers and abbreviations, participials and infinitive modifiers.

4. Traverse the subtrees and generate all possible compression alternatives using the subtree root node, then keep the top two longest sub sentences.

5. Drop the sub sentences shorter than 5 words.

5 Sentence Selection

After sentence pruning, we prepare for the final event summary generation process. In this step, we select one compressed version of the sentence from each aspect cluster. To avoid redundancy between aspects, we use Integer Linear Programming to optimize a global objective function for sentence selection. Inspired by (Sauper and Barzilay, 2009), we formulate the optimization problem based on sentence ranking information. More specifically, we would like to select exactly one compressed sentence which receives the highest possible ranking score from each aspect cluster subject to a series of constraints, such as redundancy and length. We employed lp_solver [4], an efficient mixed integer programming solver using the Branch-and-Bound algorithm to select sentences.

Original	**Compressed**
When rescue workers arrived, they said, only one of his limbs was visible.	When rescue workers arrived, only one of his limbs was visible.
Two days earlier, a massacre by two students at Columbine High, whose teams are called the Rebels, left 15 people dead and dozens wounded.	Two days earlier, a massacre by two students at Columbine High, left 15 people dead and dozens wounded.

Table 2: Example compressed sentences.

Assume that there are in total K aspects in an event topic. For each aspect j, there are in total R ranked sentences. The variables S_{jl} is a binary indicator of the sentence. That is, S_{jl}= 1 if the sentence is included in the final summary, and S_{jl} = 0 otherwise. l is the ranked position of the sentence in this aspect cluster.

Objective Function

Top ranked sentences are the most relevant corresponding to the related aspects which we want to include in the final summary. Thus we try to minimize the ranks of the sentences to improve the overall responsiveness.

$$\min(\sum_{j=1}^{K}\sum_{l=1}^{R_j} l \cdot S_{jl})$$

Exclusivity Constraints

To prevent redundancy in each aspect, we just choose one sentence from each general or specific aspect cluster. The constraint is formulated as follows:

$$\sum_{l=1}^{R_j} S_{jl} = 1 \qquad \forall j \in \{1 \dots K\}$$

[3] The parataxis relation is a relation between the main verb of a clause and other sentential elements, such as a sentential parenthetical, colon, or semicolon

[4] http://lpsolve.sourceforge.net/5.5/

Redundancy Constraints

We also want to prevent redundancy across different aspects. If sentence-similarity $sim(s_{jl}, s_{j'l'})$ between sentence s_{jl} and $s_{j'l'}$ is above 0.5, then we drop the pair and choose one sentence ranked higher from the pair otherwise. This constraint is formulated as follows:

$$(S_{jl} + S_{j'l'}) \cdot sim(s_{jl}, s_{j'l'}) \leq 0.5$$
$$\forall j, j' \in \{1 \dots K\} \forall l \in \{1 \dots R_j\} \forall l' \in \{1 \dots R_{j'}\}$$

Length Constraints

We add this constraint to ensure that the length of the final summary is limited to L words.

$$\sum_{j=1}^{K} \sum_{l=1}^{R_j} len_{jl} \cdot S_{jl} \leq L$$

where len_{jl} is the length of S_{jl}.

6 Evaluation

In order to systematically evaluate our method, we want to check (1) whether the whole system is effective, which means to quantitatively evaluate summary quality, and (2) whether individual components like clustering and compression algorithms are useful.

6.1 Data

We use TAC2010 Summarization task data set for the summary content evaluation. This data set provides 46 events. Each event falls into a predefined event topic. Each specific event includes an event statement and 20 relevant newswire articles which have been divided into 2 sets: Document Set A and Document Set B. Each document set has 10 documents, and all the documents in Set A chronologically precede the documents in Set B. We just use document Set A for our task. Assessors wrote model summaries for each event, so we can compare our automatic generated summaries with the model summaries. We combine topic related data sets together, then these data sets simultaneously annotated by our Event-aspect model. After labeling process, we run sentence ranking, compression and selection module to get final aspect-oriented summarizations.

6.2 Quality of summary

We use the ROUGE (Lin and Hovy, 2003) metric for measuring the summarization system performance. Ideally, a summarization criterion should be more recall oriented. So the average recall of ROUGE-1, ROUGE-2, ROUGE-SU4, ROUGE-W-1.2 and ROUGE-L were computed by running ROUGE-1.5.5 with stemming but no removal of stop words. We compare our method with the following four baseline methods.

Baseline 1

In this baseline, we try to compare different sentence clustering algorithms in the multi-document summarization scenario. First, we use CLUTO [5] to do K-means clustering. Then we try entity-aspect model proposed by Li et al. (2010) to do sentence clustering. Entity-aspect model is similar with "HIERSUM" content model proposed by Haghighi and Vanderwende (2009). We use the same ranking, compression, and selection components to generate aspect-oriented summaries for comparison.

Baseline 2

In this baseline, we compare our method with traditional ranking and selection summary generation framework (Erkan and Radev, 2004; Nenkova and Vanderwende, 2005) to show that our sentence clustering component is necessary in aspect-oriented summarization system. Also we want check whether sentence ranking combined with greedy based sentence selection can prevent redundancy effectively. We follow LexRank based sentence ranking combined with greedy sentence selection methods. We implement two greedy algorithms (Zhang et al., 2008; Paul et al., 2010). One is to select the top ranked sentence simultaneously by removing 10 redundant neighbor sentences from the sentence similarity graph if the summary length is less then 100 words. This is repeated until the graph cannot be partitioned. The similarity graph building threshold is 0.3, damping factor is 0.2 and error tolerance for Power Method in LexRank is 0.1. The other is to select top ranked sentences as long as the redundancy score (similarity) between a candidate sentence and

[5] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

current summary is under 0.5. This is repeated until the summary reaches a 100 word length limit.

Baseline 3

In this baseline, we compare our ILP based sentence selection with KL-divergence based sentence selection. The KL-divergence formula we use is below,

$$KL(P_S||Q_D) = \sum_w P(w) \log \frac{P(w)}{Q(w)}$$

where $P(S)$ is the empirical unigram distribution of the candidate summary S, and $Q(D)$ is the unigram distribution of document collection D. We only replaced our selection method with the KL-divergence selection method. Other parts are the same. After ranking sentences for each aspect, we add the sentence with the highest ranking score from each aspect sentence cluster as long as the KL-divergence between candidate and current summary does not decrease. This is repeated until the summary reaches a 100 word length limit. To our knowledge, this is the first work to directly compare Integer Linear Programming based sentence selection with KL-divergence based sentence selection in summarization generation framework.

Baseline 4

In this baseline, we directly compare our method with "HIERSUM" proposed by (Haghighi and Vanderwende, 2009). As in Baseline 1, we use entity-aspect model to approximate "HIERSUM" model. We replace unigram distribution of $P(w)$ in KL-divergence with learned distribution estimated by "HIERSUM" model. The KL-divergence based greedy sentence selection algorithm is similar to Baseline 3.

For fair comparison, Baselines 1, 2, 3 and 4 use the same sentence compression algorithm and have the summary length no more then 100 words. In Table 3, we show the average ROUGE recall of 46 summaries generated by our method and four baseline methods. We can see that our method gives better Rouge recall measures then the four baseline methods. For BL-1, we can see that LDA-based sentence clustering is better then k-means. For BL-2, we can see that traditional ranking plus greedy selection summary generation framework is not suitable for the aspect-oriented summarization task. More specifically, greedy-based sentence selection can not prevent redundancy effectively. BL-3 evaluation results showed that ILP-based sentence selection is better then KL-divergence selection in terms of preventing redundancy across different aspects. The measurement performance between BL-3 and BL-4 is close. They use the same KL-divergence based sentence selection, but topic model they use are different, and also BL-3 has a sentence ranking process. The Rouge recall of our method is better than BL-4. It is expected because our event-aspect model can better find the aspects and also prove that our LexRank based sentence ranking combined with ILP-based sentence selection can prevent redundancy.

Due to TAC2010 summarization community just compute ROUGE-2 and ROUGE-SU4 metrics for participants, our ROUGE-2 metric ranked 11 out of 23, ROUGE-SU4 metric ranked 12 out of 23. They use MEAD[6] as their baseline approach. The ROUGE-2 score of our approach achieve 0.06508 higher than MEAD's 0.05929. The ROUGE-SU4 score of our approach achieve 0.10146 higher than MEAD's 0.09112. Many systems that get higher performances leverage domain knowledge bases like Wikipedia or training data, but we didn't. The advantage of our method is that we generate summaries with totally unsupervised framework and this approach is domain adaptive.

6.3 Quality of aspect-oriented sentence clusters

To judge the quality of the aspect-oriented sentence clusters, we ask the human judges to group the ground truth sentences based on the aspect relatedness in each event topic. We then compute the purity of the automatically generated clusters against the human judged clusters. The results are shown in Table 4. In our experiments, we set the number of general aspect clusters A^G is 5 and specific aspect clusters A^S is 3. We can see from Table 4 that our generated aspect clusters can achieve reasonably good performance.

[6] http://www.summarization.com/mead/

Method		Rouge Average Recall				
		ROUGE-1	ROUGE-2	ROUGE-SU4	ROUGE-W-1.2	ROUGE-L
BL-1	k-means	0.21895	0.03689	0.06644	0.06683	0.19208
	entity-aspect	0.26082	0.05082	0.08286	0.08055	0.22976
BL-2	greedy 1	0.27802	0.04872	0.08302	0.08488	0.24426
	greedy 2	0.27898	0.04723	0.08275	0.08500	0.24430
BL-3	KL-Div	0.29286	0.05369	0.09117	0.08827	0.25100
BL-4	HIERSUM	0.28736	0.05502	0.08932	0.08923	0.25285
Without compression		0.30563	0.05983	0.09513	0.09468	0.25487
Our Method		0.32641	0.06508	0.10146	0.09998	0.28610

Table 3: ROUGE evaluation results on TAC2010 Summarization data sets

Category	A	Purity
Accidents and Natural Disasters	7	0.613
Attacks	8	0.658
Health and Safety	5	0.724
Endangered Resources	4	0.716
Investigations and Trials	6	0.669

Table 4: The true numbers of aspects as judged by the human annotator (A), and the purity of the clusters.

Category	Average Score
Accidents and Natural Disasters	2.4
Attacks	2.3
Health and Safety	2.6
Endangered Resources	2.5
Investigations and Trials	2.4

Table 5: The average score of each event topic.

6.4 Quality of sentence compression

To judge the quality of the dependency tree based sentence compression algorithm, we ask the human judges to choose 20 sentences from each event topic then score them. The judges follow 3-point scale to score each compressed sentence: 1 means poor, 2 means barely acceptable, and 3 means good. We then compute the average scores. The results are shown in Table 5. To evaluate the effectiveness of sentence compression component, we conduct the system without sentence compression component, then compare it with our system. In Table 3, we can see that sentence compression can improve the system performance.

7 Related Work

Our event-aspect model is related to a number of previous extensions of LDA models. Chemudugunta et al. (2007) proposed to introduce a background topic and document-specific topics. Our background and document language models are similar to theirs. However, they still treat documents as bags of words rather then sets of sentences as in our models. Titov and McDonald (2008) exploited the idea that a short paragraph within a document is likely to be about the same aspect. The way we separate words into stop words, background words, document words and aspect words bears similarity to that used in (Daumé III and Marcu, 2006; Haghighi and Vanderwende, 2009). Paul and Girju (2010) proposed a topic-aspect model for simultaneously finding topics and aspects. The most related extension is entity-aspect model proposed by Li et al. (2010). The main difference between event-aspect model and entity-aspect model is our model further consider aspect granularity and add a layer to model topic-related events.

Filippova and Strube (2008) proposed a dependency tree based sentence compression algorithm. Their approach need a large corpus to build language model for compression, whereas we prune dependency tree using grammatical rules.

Paul et al. (2010) proposed to modify LexRank algorithm using their topic-aspect model. But their task is to summarize contrastive viewpoints in opinionated text. Furthermore, they use a simple greedy approach for constructing summary.

McDonald (2007) proposed to use Integer Linear Programming framework in multi-document sum-

marization. And Sauper and Barzilay (2009) use integer linear programming framework to automatically generate Wikipedia articles. There is a fundamental difference between their method and ours. They used trained perceptron algorithm for ranking excerpts, whereas we give an extended LexRank with integer linear programming to optimize sentence selection for our aspect-oriented multi-document summarization.

8 Conclusions and Future Work

In this paper, we study the task of automatically generating aspect-oriented summary from multiple documents. We proposed an event-aspect model that can automatically cluster sentences into aspects. We then use an extension of the LexRank algorithm to rank sentences. We took advantage of the output generated by the event-aspect model to modify jumping probabilities so as to favor visiting representative sentence. We also proposed dependency tree compression algorithm to prune sentence for improving linguistic quality of the summaries. Finally we use Integer Linear Programming Framework to select aspect relevant sentences. We conducted quantitative evaluation using standard test data sets. We found that our method gave overall better ROUGE scores than four baseline methods, and the new sentence clustering and compression algorithm are robust.

There are a number of directions we plan to pursue in the future in order to improve our method. First, we can possibly apply more linguistic knowledge to improve the quality of sentence compression. Currently the sentence compression algorithm may generate meaningless subtrees. It is relatively hard to decide which clause is redundant in terms of summarization. Second, we may explore more domain knowledge to improve the quality of aspect-oriented summaries. For example, we know that the "who-affected" aspect is related to person, and "when, where" are related to Time and Location. we can import Name Entity Recognition to annotate these phrases and then help locate relevant sentences. Third, we want to extend our event-aspect model to simultaneously find topics and aspects.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC No. 60773088), the National High-tech R&D Program of China (863 Program No. 2009AA04Z106), and the Key Program of Basic Research of Shanghai Municipal S&T Commission (No. 08JC1411700).

References

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2007. Modeling general and specific aspects of documents with a probabilistic topic model. In *Advances in Neural Information Processing Systems 19*, pages 241–248.

Hal. Daumé III and Daniel. Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 305–312. Association for Computational Linguistics.

Günes. Erkan and Dragomir Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479.

K. Filippova and M. Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32. Association for Computational Linguistics.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18.

Dan Gillick, Benoit Favre, D. Hakkani-Tur, B. Bohnet, Y. Liu, and S. Xie. 2010. The icsi/utd summarization system at tac 2009. In *Proceedings of the Second Text Analysis Conference, Gaithersburg, Maryland, USA: National Institute of Standards and Technology*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl. 1):5228–5235.

A. Haghighi and L. Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on ZZZ*, pages 362–370. Association for Computational Linguistics.

Peng Li, Jing Jiang, and Yinglin Wang. 2010. Generating templates of entity summaries with an entity-aspect model and pattern mining. In *Proceedings of the Joint Conference of the 48th Annual Meeting of the ACL*. Association for Computational Linguistics.

C.Y. Lin and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. *Advances in Information Retrieval*, pages 557–564.

A. Nenkova and L. Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005-101*.

Michael J. Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *In AAAI-2010: Twenty-Fourth Conference on Artificial Intelligence*.

Michael J. Paul, ChengXiang Zhai, and Roxana Girju. 2010. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 66–76, Morristown, NJ, USA. Association for Computational Linguistics.

Christina Sauper and Regina Barzilay. 2009. Automatically generating wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 208–216, Suntec, Singapore, August. Association for Computational Linguistics.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceeding of the 17th International Conference on World Wide Web*, pages 111–120.

D. Zajic, B.J. Dorr, J. Lin, and R. Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing & Management*, 43(6):1549–1570.

Jin. Zhang, Xueqi. Cheng, and Hongbo. Xu. 2008. GSP-Summary: a graph-based sub-topic partition algorithm for summarization. In *Proceedings of the 4th Asia information retrieval conference on Information retrieval technology*, pages 321–334. Springer-Verlag.

Syntax-Based Grammaticality Improvement using CCG and Guided Search

Yue Zhang
University of Cambridge
Computer Laboratory
yue.zhang@cl.cam.ac.uk

Stephen Clark
University of Cambridge
Computer Laboratory
stephen.clark@cl.cam.ac.uk

Abstract

Machine-produced text often lacks grammaticality and fluency. This paper studies grammaticality improvement using a syntax-based algorithm based on CCG. The goal of the search problem is to find an optimal parse tree among all that can be constructed through selection and ordering of the input words. The search problem, which is significantly harder than parsing, is solved by guided learning for best-first search. In a standard word ordering task, our system gives a BLEU score of 40.1, higher than the previous result of 33.7 achieved by a dependency-based system.

1 Introduction

Machine-produced text, such as SMT output, often lacks grammaticality and fluency, especially when using n-gram language modelling (Knight, 2007). Recent efforts have been made to improve grammaticality using local language models (Blackwood et al., 2010) and global dependency structures (Wan et al., 2009). We study grammaticality improvement using a syntax-based system.

The task is effectively a text-to-text generation problem where the goal is to produce a grammatical sentence from an ungrammatical and fragmentary input. The input can range from a bag-of-words (Wan et al., 2009) to a fully-ordered sentence (Blackwood et al., 2010). A general form of the problem is to construct a grammatical sentence from a set of un-ordered input words. However, in cases where the base system produces fluent subsequences within the sentence, constraints on the choice and order of certain words can be fed to the grammaticality improvement system. The input may also include words beyond the output of the base system, e.g. extra words from the SMT lattice, so that content word insertion and deletion can be performed implicity via word selection.

We study the above task using CCG (Steedman, 2000). The main challenge is the search problem, which is to find an optimal parse tree among all that can be constructed with any word choice and order from the set of input words. We use an approximate best-first algorithm, guided by learning, to tackle the more-than-factorial complexity. Beam-search is used to control the volume of accepted hypotheses, so that only a very small portion of the whole search space is explored. The search algorithm is guided by perceptron training, which ensures that the explored path in the search space consists of highly probable hypotheses. This framework of best-first search guided by learning is a general contribution of the paper, which could be applied to problems outside grammaticality improvement.

We evaluate our system using the generation task of word-order recovery, which is to recover the original word order of a fully scrambled input sentence (Bangalore et al., 2000; Wan et al., 2009). This problem is an instance of our general task formulation, but without any input constraints, or content word selection (since all input words are used). It is straightforward to use this task to evaluate our system and compare with existing approaches. Our system gave 40.1 BLEU score, higher than the dependency-based system of Wan et al. (2009), for which a BLEU score of 33.7 was reported.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1147–1157,
Edinburgh, Scotland, UK, July 27–31, 2011.

2 The Grammar

Combinatory Categorial Grammar (CCG; Steedman (2000)) is a lexicalized grammar formalism, which associates words with lexical categories. Lexical categories are detailed grammatical labels, typically expressing subcategorisation information. CCG, and parsing with CCG, has been described in detail elsewhere (Clark and Curran, 2007; Hockenmaier, 2003); here we provide only a short description.

During CCG parsing, adjacent categories are combined using CCG's combinatory rules. For example, a verb phrase in English ($S\backslash NP$) can combine with an NP to its left:

$$NP \;\; S\backslash NP \Rightarrow S$$

In addition to binary rule instances, such as the one above, there are also unary rules which operate on a single category in order to change its type. For example, forward type-raising can change a subject NP into a complex category looking to the right for a verb phrase:

$$NP \Rightarrow S/(S\backslash NP)$$

Following Hockenmaier (2003), we extract the grammar by reading rule instances directly from the derivations in CCGbank (Hockenmaier and Steedman, 2007), rather than defining the combinatory rule schema manually as in Clark and Curran (2007).

3 The Search Algorithm

The input to the search algorithm is a set of words, each word having a count that specifies the maximum number of times it can appear in the output. Typically, most input words can occur only once in the output. However, punctuation marks and function words can be given a higher count. Depending on the fluency of the base output (e.g. the output of the base SMT system), some constraints can be given to specific input words, limiting their order or identifying them as an atomic phrase, for example.

The goal of the search algorithm is to find an optimal parse tree (including the surface string) among all that can be constructed via selecting and ordering a subset of words from the input multiset. The complexity of this problem is much higher than a typical parsing problem, since there is an exponential number of word choices for the output sentence, each with a factorial number of orderings. Moreover, dynamic programming packing for parsers, such as a CKY chart, is not applicable, because of the lack of a fixed word order.

We perform approximate search using a best-first algorithm. Starting from single words, candidate parses are constructed bottom-up. Similar to a best-first parser (Caraballo and Charniak, 1998), the highest scored hypothesis is expanded first. A hypothesis is expanded by applying CCG unary rules to the hypothesis, or by combining the hypothesis with existing hypotheses using CCG binary rules.

We use beam search to control the number of accepted hypotheses, so that the computational complexity of expanding each hypothesis is linear in the size of the beam. Since there is no guarantee that a goal hypothesis will be found in polynomial time, we apply a robustness mechanism (Riezler et al., 2002; White, 2004), and construct a default output when no goal hypothesis is found within a time limit.

3.1 Data Structures

Edges are the basic structures that represent hypotheses. Each edge is a CCG constituent, spanning a sequence of words. Similar to partial parses in a typical chart parser, edges have recursive structures. Depending on the number of subedges, edges can be classified into *leaf edges*, *unary edges* and *binary edges*. Leaf edges, which represent input words, are constructed first in the search process. Existing edges are expanded to generate new edges via unary and binary CCG rules. An edge that meets the output criteria is called a *goal edge*. In the experiments of this paper, we define a goal edge as one that includes all input words the correct number of times.

The *signature* of an edge consists of the category label, surface string and head word of the constituent. Two edges are *equivalent* if they share the same signature. Given our feature definitions, a lower scoring edge with the same signature as a higher scoring edge cannot be part of the highest scoring derivation.

The number of words in the surface string of an edge is called the *size* of the edge. Other important substructures of an edge include a bitvector and an array, which stores the indices of the input words that the edge contains. Before two edges are combined using a binary CCG rule, an *input check* is per-

formed to make sure that the total count for a word from the two edges does not exceed the count for that word in the input. Intuitively, an edge can record the count of each unique input word it contains, and perform the input check in linear time. However, since most input words typically occur once, they can be indexed and represented by a bitvector, which allows a constant time input check. The few multiple-occurrence words are stored in a count array.

In the best-first process, edges to be expanded are ordered by their scores, and stored in an *agenda*. Edges that have been expanded are stored in a *chart*. There are many ways in which edges could be ordered and compared. Here the chart is organised as a set of beams, each containing a fixed number of edges with a particular size. This is similar to typical decoding algorithms for phrase-based SMT (Koehn, 2010). In each beam, edges are ordered by their scores, and low score edges are pruned. In addition to pruning by the beam, only the highest scored edge is kept among all that share the same signature.

3.2 The Search Process

Figure 1 shows pseudocode for the search algorithm. During initialization, the agenda (a) and chart (c) are cleared. All candidate lexical categories are assigned to each input word, and the resulting leaf edges are put onto the agenda.

In the main loop, the best edge (e) is popped from the agenda. If e is a goal hypothesis, it is appended to a list of goals (*goal*), and the loop is continued without e being expanded. If e or any equivalent edge $\tilde{e}$ of e is already in the chart, the loop continues without expanding e. It can be proved that any edge in the chart must have been combined with $\tilde{e}$, and therefore the expansion of e is unnecessary.

Edge e is first expanded by applying unary rules, and any new edges are put into a list (*new*). Next, e is matched against each existing edge $\tilde{e}$ in the chart. e and $\tilde{e}$ can be combined if they pass the input check, and there is a binary rule in which the constituents are combined. e and $\tilde{e}$ are combined in both possible orders, and any resulting edge is added to *new*.

At the end of each loop, edges from *new* are added to the agenda, and *new* is cleared. The loop continues until a stopping criterion is met. A typical stopping condition is that *goal* contains N goal edges.

```
a ← INITAGENDA(input)
c ← INITCHART()
new ← []
goal ← []
while not STOP(goal, time):
  e ← POPBEST(a)
  if GOALTEST(e)
    APPEND(goal, e)
    continue
  for ẽ in c:
    if EQUIV(ẽ, e):
      continue
  for e′ in UNARY(e, grammar):
    APPEND(new, e′)
  for ẽ in c:
    if CANCOMBINE(e, ẽ):
      e′ ← BINARY(e, ẽ, grammar)
      APPEND(new, e′)
    if CANCOMBINE(ẽ, e):
      e′ ← BINARY(ẽ, e, grammar)
      APPEND(new, e′)
  for e′ in new:
    ADD(a, e′)
  ADD(c, e)
  new ← []
```

Figure 1: The search algorithm.

We set N to 1 in our experiments. For practical reasons we also include a timeout stopping condition. If no goal edges are found before the timeout is reached, a default output is constructed by the following procedure. First, if any two edges in the chart pass the input check, and the words they contain constitute the full input set, they are concatenated to form an output string. Second, when no two edges in the chart meet the above condition, the largest edge $\tilde{e}$ in the chart is chosen. Then edges in the chart are iterated over in the larger first order, with any edge that passes the input check with $\tilde{e}$ concatenated with $\tilde{e}$ and $\tilde{e}$ updated. The final $\tilde{e}$, which can be shorter than the input, is taken as the default output.

4 Model and Features

We use a discriminative linear model to score edges, where the score of an edge e is calculated using the global feature vector $\Phi(e)$ and the parameter vector

$\vec{w}$ of the model.

$$\text{SCORE}(e) = \Phi(e) \cdot \vec{w}$$

$\Phi(e)$ represents the counts of individual features of e. It is computed incrementally as the edge is built. At each constituent level, the incremental feature vector is extracted according to the feature templates from Table 1, and we use the term *constituent level vector* ϕ to refer to it. So for any edge e, $\phi(e)$ consists of features from the top rule of the hierarchical structure of e. $\Phi(e)$ can be written as the sum of $\phi(e')$ of all recursive subedges e' of e, including e itself:

$$\Phi(e) = \sum_{e' \in e} \phi(e')$$

The parameter update in Section 5 is in terms of constituent level vectors.

The features in Table 1 represent patterns including the constituent label; the head word of the constituent; the size of the constituent; word, POS and lexical category N-grams resulting from a binary combination; and the unary and binary rules by which the constituent is constructed. They can be classified roughly into "parsing" features (those about the parse structure, such as the binary rule) and "generation features" (those about the surface string, such as word bigrams), although some features, such as "rule + head word + non-head word", contain both types of information.

condition	feature
all edges	constituent + size
	constituent + head word
	constituent + size + head word
	constituent + head POS
size > 1	constituent + leftmost word
	constituent + rightmost word
	consti. + leftmost POS bigram
	consti. + rightmost POS bigram
	consti. + lmost POS + rmost POS
binary edges	the binary rule
	the binary rule + head word
	rule + head word + non-head word
	bigrams resulting from combination
	POS bigrams resulting from combi.
	word trigrams resulting from combi.
	POS trigrams resulting from combi.
	resulting lexical categary trigrams
	resulting word + POS bigrams
	resulting POS + word bigrams
	resulting POS + word + POS trigrams
unary edges	unary rule
	unary rule + headw

Table 1: Feature template definitions.

5 The Learning Algorithm

The statistical model plays two important roles in our system. First, as in typical statistical systems, it is expected to give a higher score to a more correct hypothesis. Second, it is also crucial to the speed of the search algorithm, since the best-first mechanism relies on a model to find goal hypotheses efficiently. As an indication of the impact of the model on efficiency, if the model parameters are set to all zeros, the search algorithm cannot find a result for the first sentence in the development data within two hours.

We perform training on a corpus of CCG derivations, where constituents in a gold-standard derivation serve as gold edges. The training algorithm runs the decoder on each training example, updating the model when necessary, until the gold goal edge is recovered. We use the perceptron (Rosenblatt, 1958) to perform parameter updates. The traditional perceptron has been adapted to structural prediction (Collins, 2002) and search optimization problems (Daumé III and Marcu, 2005; Shen et al., 2007). Our training algorithm can be viewed as an adaptation of the perceptron to our best-first framework for search efficiency and accuracy.

We choose to update parameters as soon as the best edge from the agenda is not a gold-standard edge. The intuition is that all gold edges are forced to be above all non-gold edges on the agenda. This is a strong precondition for parameter updates. An alternative is to update when a gold-standard edge falls off the chart, which corresponds to the precondition for parameter updates of Daumé III and Marcu (2005). However, due to the complexity of our search task, we found that reasonable training efficiency cannot be achieved by the weaker alternatives. Our updates lead both to correctness (edges in the chart are correct) and efficiency (correct edges are found at the first possible opportunity).

During a perceptron update, an incorrect prediction, corresponding to the current best edge in the agenda, is penalized, and the corresponding gold edge is rewarded. However, in our scenario it is not obvious what the corresponding gold edge should be, and there are many ways in which the gold edge could be defined. We investigated a number of alternatives, for example trying to find the "best match" for the incorrect prediction. In practice we found that the simple strategy of selecting the lowest scored gold-standard edge in the agenda was effective, and the results presented in this paper are based on this method.

After an update, there are at least two alternative methods to continue. The first is to reinitialize the agenda and chart using the new model, and continue until the current training example is correctly predicted. This method is called *aggressive training* (Shen et al., 2007). In order to achieve reasonable efficiency, we adopt a second approach, which is to continue training without reinitializing the agenda and chart. Instead, only edges from the top of the agenda down to the lowest-scoring gold-standard edge are given new scores according to the new parameters.

Figure 2 shows pseudocode for the learning algorithm applied to one training example. The initialization is identical to the test search, except that the list of goal edges is not maintained. In the main loop, the best edge e is popped off the agenda. If it is the gold goal edge, the training for this sentence finishes. If e is not a gold edge, parameter updates are performed and the loop is continued with e being discarded. Only gold edges are pushed onto the chart throughout the training process.

When updating parameters, the current non-gold edge (e) is used as the negative example, and the smallest gold edge in the agenda (*minGold*) is used as the corresponding positive example. The model parameters are updated by adding the constituent level feature vector (see Section 4) of *minGold*, and subtracting the constituent level feature vector of e. Note that we do not use the global feature vector in the update, since only the constituent level parameter vectors are compatible for edges with different sizes. After a parameter update, edges are rescored from the top of the agenda down to *minGold*.

The training algorithm iterates through all training examples N times, and the final parameter vector is used as the model. In our experiments, N is chosen according to results on development data.

```
a ← INITAGENDA(input)
c ← INITCHART()
new ← []
while true:
  e ← POPBEST(a)
  if GOLD(e) and GOALTEST(e):
    return
  if not GOLD(e):
    popped ← []
    n ← 0
    while n < GOLDCOUNT(a):
      ẽ ← POPBEST(a)
      APPEND(popped, ẽ)
      if GOLD(ẽ):
        minGold ← ẽ
        n ← n + 1
    w⃗ ← w⃗ − φ(e) + φ(minGold)
    for ẽ in popped:
      RECOMPUTESCORE(ẽ)
      ADD(a, ẽ)
    for ẽ in c:
      RECOMPUTESCORE(ẽ)
    continue
  for e′ in UNARY(e, grammar):
    APPEND(new, e′)
  for ẽ in c:
    if CANCOMBINE(e, ẽ):
      e′ ← BINARY(e, ẽ, grammar)
      APPEND(new, e′)
    if CANCOMBINE(ẽ, e):
      e′ ← BINARY(ẽ, e, grammar)
      APPEND(new, e′)
  for e′ in new:
    ADD(a, e′)
  ADD(c, e)
  new ← []
```

Figure 2: The learning algorithm.

6 Experiments

We use CCGBank (Hockenmaier and Steedman, 2007) for experimental data. CCGbank is the CCG version of the Penn Treebank. Sections 02–21 are

used for training, section 00 is used for development and section 23 for the final test.

Original sentences from CCGBank are transformed into bags of words, with sequence information removed, and passed to our system as input data. The system outputs are compared to the original sentences for evaluation. Following Wan et al. (2009), we use the BLEU metric (Papineni et al., 2002) for string comparison. Whilst BLEU is not an ideal measure of fluency or grammaticality, being based on n-gram precision, it is currently widely used for automatic evaluation and allows us to compare directly with existing work (Wan et al., 2009).

In addition to the surface string, our system also produces the CCG parse given an input bag of words. The quality of the parse tree can reflect both the grammaticality of the surface string and the quality of the trained grammar model. However, there is no direct way to automatically evaluate parse trees since output word choice and order can be different from the gold-standard. Instead, we indirectly measure parse quality by calculating the precision of CCG lexical categories. Since CCG lexical categories contain so much syntactic information, they provide a useful measure of parse quality. Again because the word order can be different, we turn both the output and the gold-standard into a bag of word/category pairs, and calculate the percentage of matched pairs as the lexical category precision.

For fair comparison with Wan et al. (2009), we keep base NPs as atomic units when preparing the input. Wan et al. (2009) used base NPs from Penn Treebank annotation, while we extract base NPs from the CCGBbank by taking as base NPs the NPs that do not recursively contain other NPs. These base NPs mostly correspond to the base NPs from the Penn Treebank. In the training data, there are 242,813 Penn Treebank base NPs with an average size of 1.09, and 216,670 CCGBank base NPs with an average size of 1.19.

6.1 Development Tests

Table 2 shows a set of development experiment results after one training iteration. Three different methods of assigning lexical categories are used. The first ("dictionary") is to assign all possible lexical categories to each input word from the dictionary. The lexical category dictionary is built using the training sections of CCGBank. For each word occurring more than 20 times in the corpus, the dictionary has an entry with all lexical categories the word has been seen with. For the rest of the words, the dictionary maintains an entry for each POS which contains all lexical categories it has been seen with. There are on average 26.8 different categories for each input word by this method.

Method	Timeout	BLEU	Length ratio	Timeout ratio
dictionary	0.5s	34.98	84.02	62.26
	1s	35.40	85.66	57.87
	5s	36.27	89.05	45.79
	10s	36.45	89.13	42,13
	50s	37.07	92.52	32.41
$\beta = 0.0001$	0.5s	36.54	84.26	66.07
	1s	37.50	86.69	58.22
	5s	38.75	90.15	43.23
	10s	39.14	91.35	38.36
	50s	39.58	93.09	30.53
$\beta = 0.075$	0.5s	40.87	85.66	61.27
	1s	42.04	87.99	53.11
	5s	43.99	91.20	40.30
	10s	44.23	92.14	35.70
	50s	45.08	93.70	29.43

Table 2: Development tests using various levels of lexical categories and timeouts, after one training iteration.

In practice, it is often unnecessary to leave lexical category disambiguation completely to the grammaticality improvement system. When it is reasonable to assume that the input sentence for the grammaticality improvement system is sufficiently fluent, a list of candidate lexical categories can be assigned automatically to each word via supertagging (Clark and Curran, 2007) on the input sequence. We use the C&C supertagger[1] to assign a set of probable lexical categories to each input word using the gold-standard order. When the input is noisy, the accuracy of a supertagger tends to be lower than when the input is grammatical. One way to address this problem is to allow the supertagger to produce a larger list of possible supertags for each input word, and leave the ambiguity to the grammatical improvement system. We simulate the noisy input situation by using

[1] http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Download.

	Precision
dictionary	58.5%
$\beta = 0.0001$	59.7%
$\beta = 0.075\%$	77.0%

Table 3: Lexical category accuracies. Timeout = 5s. 1 training iteration.

a small probability cutoff (β) value in the supertagger, and supertag correctly ordered input sentences before breaking them into bags of words. With a β value of 0.0001, there are 5.4 lexical categories for each input word in the development test (which is smaller than the dictionary case).

The average number of lexical categories per word drops to 1.3 when β equals 0.075, which is the value used for parsing newspaper text in Clark and Curran (2007). We include this β in our experiments to compare the effect of different β levels.

The table shows that the BLEU score of the grammaticality improvement system is higher when a super tagger is used, and the higher the β value, the better the BLEU score. In practice, the β value should be set in accordance with the lack of grammaticality and fluency in the input. The dictionary method can be used when the output is extremely unreliable, while a small beta value can be used if the output is almost fluent.

Due to the default output mechanism on timeout, the system can sometimes fail to produce sentences that cover all input words. We choose five different timeout settings between 0.5s to 50s, and compare the speed/quality tradeoff. In addition to BLEU, we report the percentage of timeouts and the ratio of the sum of all output sentence lengths to the sum of all input sentence lengths.

When the timeout value increases, the BLEU score generally increases. The main effect of a larger timeout is the increased possibility of a complete sentence being found. As the time increases from 0.5s to 50s using the dictionary method, for example, the average output sentence length increases from 84% of the input length to 93%.

Table 3 shows the lexical category accuracies using the dictionary, and supertagger with different β levels. The timeout limit is set to 5 seconds. As the lexical category ambiguity decreases, the accuracy increases. The best lexical category accuracy of 77% is achieved when using a supertagger with a β level 0.075, the level for which the least lexical category disambiguation is required. However, compared to the 93% lexical category accuracy of a CCG parser (Clark and Curran, 2007), which also uses a β level of 0.075 for the majority of sentences, the accuracy of our grammaticality improvement system is much lower. The lower score reflects the lower quality of the parse trees produced by our system. Besides the difference in the algorithms themselves, one important reason is the much higher complexity of our search problem.

Length	dictionary	$\beta = 0.0001$	$\beta = 0.075$
≤ 5	75.65	89.42	92.64
≤ 10	57.74	66.00	78.54
≤ 20	42.44	48.89	58.23
≤ 40	37.48	40.32	46.00
≤ 80	36.50	39.01	44.26
all	36.27	38.75	43.99

Table 4: BLEU scores measured on different lengths on development data. Timeout = 5s. 1 training iteration.

Table 4 shows the BLEU scores measured by different sizes of input. We also give some example output sentences in Figure 3. It can be seen from the table that the BLEU scores are higher when the size of input is smaller. For sentences shorter than 20 words, our system generally produces reasonably fluent and grammatical outputs. For longer sentences, the grammaticality drops. There are three possible reasons. First, larger constituents require more steps to construct. The model and search algorithm face many more ambiguities, and error propagation is more severe. Second, the search algorithm often fails to find a goal hypothesis before timeout, and a default output that is less grammatical than a complete constituent is constructed. Long sentences have comparatively more input words uncovered in the output. Third, the upper bound is not 100, and presumably lower for longer sentences, because there are many ways to generate a grammatical sentence given a bag of words. For example, the bag { cats, chase, dogs } can produce two equally fluent and grammatical sentences.

The relatively low score for long sentences is un-

(dictionary) our products There is no asbestos in now .
($\beta = 0.0001$) in our products now There is no asbestos .
($\beta = 0.075$) There is no asbestos in our products now .

(dictionary) No price for the new shares has been set .
(both β) No price has been set for the new shares .

(all) Federal Data Corp. got a $ 29.4 million Air Force contract for intelligence data handling .

(dictionary) was a nonexecutive director of Rudolph Agnew and former chairman of Consolidated Gold Fields PLC , this British industrial conglomerate , 55 years old . named
($\beta = 0.0001$) old Consolidated Gold Fields PLC , was named 55 years , former chairman of Rudolph Agnew and a nonexecutive director of this British industrial conglomerate .
($\beta = 0.075$) Consolidated Gold Fields PLC , 55 years old , was named former chairman of Rudolph Agnew and a nonexecutive director of this British industrial conglomerate .

(dictionary) McDermott International Inc. said its Babcock & Wilcox unit completed the sale of its Bailey Controls Operations for Finmeccanica S.p . A. to $ 295 million .
($\beta = 0.0001$) $ 295 million McDermott International Inc. for the sale of its Babcock & Wilcox unit said its Bailey Controls Operations completed to Finmeccanica S.p . A. .
($\beta = 0.075$) McDermott International Inc. said its Bailey Controls Operations completed the sale of Finmeccanica S.p . A. for its Babcock & Wilcox unit to $ 295 million .

Figure 3: Example outputs on development data.

likely to be such a problem in practice, because the base system (e.g. an SMT system) is likely to produce sentences with locally fluent subsequences. When fluent local phrases in the input are treated as atomic units, the effective sentence length is shorter.

All the above development experiments were performed using only one training iteration. Figure 4 shows the effect of different numbers of training iterations. For the final test, based on the graphs in Figure 4, we chose the training iterations to be 8, 6 and 4 for the dictionary, $\beta = 0.0001$ and $\beta = 0.075$ methods, respectively.

6.2 Final Accuracies

Table 5 shows the final results of our system, together with the MST-based ("Wan 2009 CLE") and assignment-based ("Wan 2009 AB") systems of Wan et al. (2009). Our system outperforms the dependency grammar-based systems, and using a supertagger with small β value produces the best BLEU. Note that through the use of a supertagger, we are no longer assuming that the input is a bag of words without any order, and therefore only the dictionary results are directly comparable with Wan et al. (2009)[2].

	BLEU
Wan 2009 CLE	26.8
Wan 2009 AB	33.7
This paper dictionary	40.1
This paper $\beta = 0.0001$	43.2
This paper $\beta = 0.075$	50.1

Table 5: Final accuracies.

[2]We also follow Wan et al. (2009) by assuming each word is associated with its POS tag.

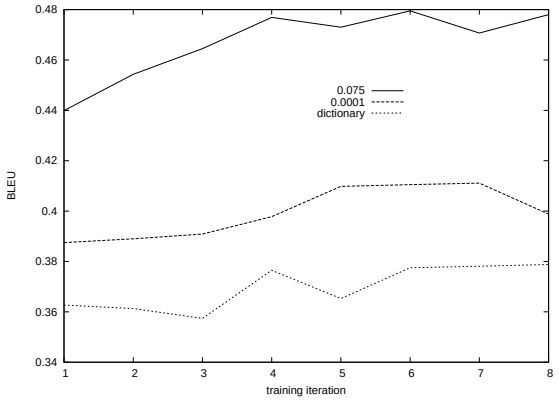


Figure 4: The effect of training iterations.

7 Related Work

Both Wan et al. (2009) and our system use approximate search to solve the problem of input word ordering. There are three differences. First, Wan et al. use a dependency grammar to model grammaticality, while we use CCG. Compared to dependency trees, CCG has stronger category constraints on the parse structure. Moreover, CCG allows us to reduce the ambiguity level of the search algorithm through the assignment of possible lexical categories to input words, which is useful when the input has a basic degree of fluency, as is often the case in a grammaticality improvement task.

Second, we use learning to optimise search in order to explore a large search space. In contrast, Wan et al. break the search problem into a sequence of sub tasks and use greedy search to connect them. Finally, in addition to ordering, our algorithm further allows word selection. This gives our system the flexibility to support word insertion and deletion.

White (2004) describes a system that performs CCG realization using best-first search. The search process of our algorithm is similar to his work. The problem we solve is different from realization, which takes an input in logical form and produces a corresponding sentence. Without constraints, the word order ambiguities can be much larger with a bag of words, and we use learning to guide our search algorithm. Espinosa et al. (2008) apply hypertagging to logical forms to assign lexical categories for realization. White and Rajkumar (2009) further use perceptron reranking on N-best outputs to improve the quality.

The use of perceptron learning to improve search has been proposed in guided learning for easy-first search (Shen et al., 2007) and LaSO (Daumé III and Marcu, 2005). LaSO is a general framework for various search strategies. Our learning algorithm is similar to LaSO with best-first inference, but the parameter updates are different. In particular, LaSO updates parameters when all correct hypotheses are lost, but our algorithm makes an update as soon as the top item from the agenda is incorrect. Our algorithm updates the parameters using a stronger precondition, because of the large search space. Given an incorrect hypothesis, LaSO finds the corresponding gold hypothesis for perceptron update by constructing its correct sibling. In contrast, our algorithm takes the lowest scored gold hypothesis currently in the agenda to avoid updating parameters for hypotheses that may have not been constructed.

Our parameter update strategy is closer to the guided learning mechanism for the easy-first algorithm of Shen et al. (2007), which maintains a queue of hypotheses during search, and performs learning to ensure that the highest scored hypothesis in the queue is correct. However, in easy-first search, hypotheses from the queue are ranked by the score of their next action, rather than the hypothesis score. Moreover, Shen et al. use aggressive learning and regenerate the queue after each update, but we perform non-agressive learning, which is faster and is more feasible for our complex search space. Similar methods to Shen et al. (2007) have also been used in Shen and Joshi (2008) and Goldberg and Elhadad (2010).

8 Conclusion

We proposed a grammaticality improvement system using CCG, and evaluated it using a standard input word ordering task. Our system gave higher BLEU scores than the dependency-based system of Wan et al. (2009). We showed that the complex search problem can be solved effectively using guided learning for best-first search.

Potential improvements to our system can be made in several areas. First, a large scale language model can be incorporated into our model in the search algorithm, or through reranking. Second, a heuristic future cost (e.g. Varges and Mel-

lish (2010)) can be considered for each hypothesis so that it also considers the words that have not been used, leading to better search. Future work also includes integration with an SMT system, where content word selection will be applicable.

Acknowledgements

We thank Graeme Blackwood, Bill Byrne, Adrià de Gispert, Stephen Wan and the anonymous reviewers for their discussions and suggestions. Yue Zhang and Stephen Clark are supported by the European Union Seventh Framework Programme (FP7-ICT-2009-4) under grant agreement no. 247762.

References

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of the First International Natural Language Generation Conference (INLG2000), Mitzpe*, pages 1–8.

Graeme Blackwood, Adrià de Gispert, and William Byrne. 2010. Fluency constraints for minimum bayes-risk decoding of statistical machine translation lattices. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 71–79, Beijing, China, August. Coling 2010 Organizing Committee.

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Comput. Linguist.*, 24:275–298, June.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8, Philadelphia, USA.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*, pages 169–176.

Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio, June. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California, June. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Julia Hockenmaier. 2003. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Meeting of the ACL*, pages 359–366, Sapporo, Japan.

Kevin Knight. 2007. Automatic language translation generation help needs badly. In *MT Summit XI Workshop on Using Corpora for NLG: Keynote Address*.

Phillip Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. III Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Libin Shen and Aravind Joshi. 2008. LTAG dependency parsing with bidirectional incremental construction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 495–504, Honolulu, Hawaii, October. Association for Computational Linguistics.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of ACL*, pages 760–767, Prague, Czech Republic, June.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, Mass.

Sebastian Varges and Chris Mellish. 2010. Instance-based natural language generation. *Natural Language Engineering*, 16(3):309–346.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2009. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model.

In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 852–860, Athens, Greece, March. Association for Computational Linguistics.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore, August. Association for Computational Linguistics.

Michael White. 2004. Reining in CCG chart realization. In *Proc. INLG-04*, pages 182–191.

Generating Subsequent Reference in Shared Visual Scenes: Computation vs. Re-Use

Jette Viethen[1,2]
jette.viethen@mq.edu.au
[1]TiCC
University of Tilburg
Tilburg, The Netherlands

Robert Dale[2]
robert.dale@mq.edu.au
[2]Centre for Language Technology
Macquarie University
Sydney, Australia

Markus Guhe[3]
m.guhe@ed.ac.uk
[3]School of Informatics
University of Edinburgh
Edinburgh, UK

Abstract

Traditional computational approaches to referring expression generation operate in a deliberate manner, choosing the attributes to be included on the basis of their ability to distinguish the intended referent from its distractors. However, work in psycholinguistics suggests that speakers align their referring expressions with those used previously in the discourse, implying less deliberate choice and more subconscious reuse. This raises the question as to which is a more accurate characterisation of what people do. Using a corpus of dialogues containing 16,358 referring expressions, we explore this question via the generation of subsequent references in shared visual scenes. We use a machine learning approach to referring expression generation and demonstrate that incorporating features that correspond to the computational tradition does not match human referring behaviour as well as using features corresponding to the process of alignment. The results support the view that the traditional model of referring expression generation that is widely assumed in work on natural language generation may not in fact be correct; our analysis may also help explain the oft-observed redundancy found in human-produced referring expressions.

1 Introduction

Computational work on referring expression generation (REG) has an extensive history, and a wide variety of algorithms have been proposed, dealing with various facets of what is recognised to be a complex problem. Almost all of this work sees the task as being concerned with choosing those attributes of an intended referent that distinguish it from the other entities with which it might be confused (see, for example, Dale (1989), Dale and Reiter (1995), Krahmer et al. (2003), van Deemter and Krahmer (2007), Gardent and Striegnitz (2007)). Independently, an alternative way of thinking about reference has arisen within the psycholinguistics community: there is now a long tradition of work that explores how a dialogue participant's forms of reference are influenced by those previously used for a given entity. Most recently, this line of work has been discussed in terms of the notions of *alignment* (Pickering and Garrod, 2004) and *conceptual pacts* (Clark and Wilkes-Gibbs, 1986; Brennan and Clark, 1996).

We suspect that neither approach tells the full story, and so we are interested in exploring whether the two perspectives should be integrated. Using a large corpus of referring expressions in task-oriented dialogues, this paper presents a machine learning approach that allows us to combine features corresponding to the two perspectives. Our results show that models based on the alignment perspective outperform models based on traditional REG considerations, as well as a number of simpler baselines.

The paper is structured as follows. In Section 2, we outline the two perspectives on subsequent reference, and summarise related work. In Section 3, we describe the iMAP Corpus and the referring expressions it contains. In Section 4, we describe the approach we take to learning models of referential behaviour using this data, and in Section 5 we discuss the results of a number of experiments based

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1158–1167,
Edinburgh, Scotland, UK, July 27–31, 2011.

on this approach, followed by an error analysis in Section 6. Section 7 draws some conclusions and discusses future work.

2 Related Work

2.1 The Algorithmic Approach

We use the term *algorithmic approach* here to refer to the perspective that is common to the considerable body of work within computational linguistics on the problem of referring expression generation developed over the last 20 years. Much of this work takes as its starting point the characterisation of the problem expressed in (Dale, 1989). This work has focused on the design of algorithms which take into account the context of reference in order to decide what properties of an entity should be mentioned in order to distinguish that entity from others with which it might be confused. Early work was concerned with *subsequent* reference in *discourse*, inspired by Grosz and Sidner's (1986) observations on how the attentional structure of a discourse made particular referents accessible at any given point. More recently, attention has shifted to *initial* reference in *visual* domains, driven in large part by the availability of the TUNA dataset and the shared tasks that make use of it (Gatt et al., 2008). The construction of *distinguishing descriptions* has consistently been a key consideration in this body of work.

Scenarios that require the generation of references in multi-turn dialogues that concern visual scenes are likely to be among the first where we can expect computational approaches to referring expression generation to be practically useful. Surprisingly, however, the more recent work on initial reference in visual domains and the earlier work on subsequent reference in discourse remain somewhat distinct and separate from each other, despite much the same algorithms having been used in both. There is very little work that brings these two strands together by looking at both initial and subsequent references in dialogues that concern visual scenes. An exception here is the machine learning approach developed by Stoia et al. (2006), who aimed at building a dialogue system for a situated agent giving instructions in a virtual 3D world. However, their approach was concerned with choosing the *type* of reference to use (definite or indefinite, pronominal, bare or modified head noun), and not with the *content* of the reference; and their data set consisted of only 1242 referring expressions.

2.2 The Alignment Approach

Meanwhile, starting with the early work of Carroll (1980), a quite distinct strand of research in psycholinguistics has explored how a speaker's form of reference to an entity is impacted by the way that entity has been previously referred to in the discourse or dialogue. The general idea behind what we will call the *alignment approach* is that a conversational participant will often adopt the same semantic, syntactic and lexical alternatives as the other party in a dialogue. This perspective is most strongly associated with the work of Pickering and Garrod (2004). With respect to reference in particular, speakers are said to form *conceptual pacts* in their use of language (Clark and Wilkes-Gibbs, 1986; Brennan and Clark, 1996). Although there is disagreement about the exact mechanisms that enable alignment and conceptual pacts, the implication of much of this work is that one speaker introduces an entity by means of some description, and then (perhaps after some negotiation) both conversational participants share this form of reference, or a form of reference derived from it, when they subsequently refer to that entity.

Recent work by Goudbeek and Krahmer (2010) supports the view that subconscious alignment does indeed take place at the level of content selection for referring expressions. The participants in their study were more likely to use a dispreferred attribute to describe a target referent if this attribute had recently been used in a description by a confederate.

There is some work within natural language generation that attempts to model the process of alignment (Buschmeier et al., 2009; Janarthanam and Lemon, 2009), but this is predominantly concerned with what we might think of as the 'lexical perspective', focussing on lexical choice rather than the selection of appropriate semantic content for distinguishing descriptions.

2.3 Combined Models

This paper is not the first to look at how the algorithmic approach and the alignment approach might be integrated in REG. An early machine learning ap-

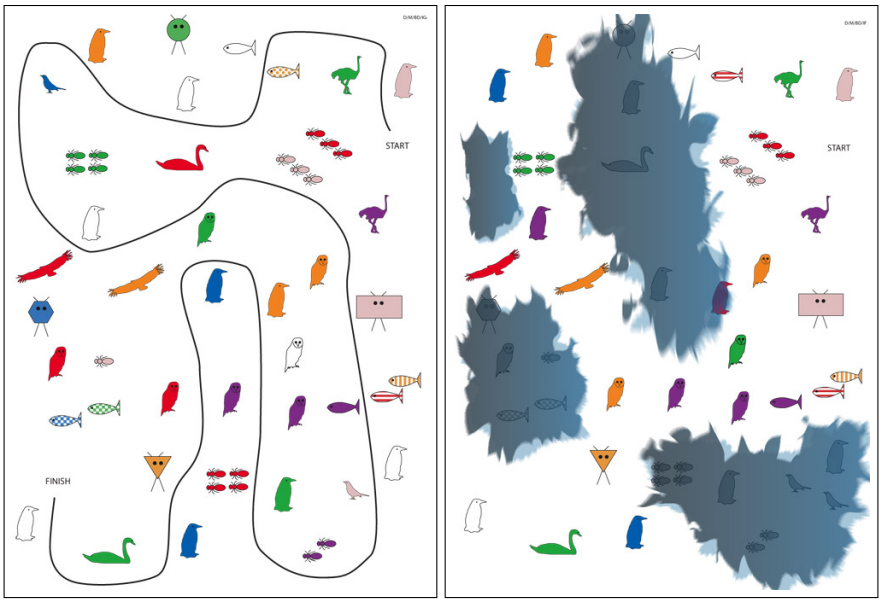


Figure 1: An example pair of maps.

proach to content selection was presented by Jordan and Walker (2000; 2005); they were also interested in an exploration of the validity of different psycholinguistic models of reference production, including Grosz and Sidner's (1986) model of discourse structure, the conceptual pacts model of Clark and colleagues, and the intentional influences model developed by Jordan (2000). However, their data set consists of only 393 referring expressions, compared to our 16,358, and these expressions had functions other than identification; most importantly, the entities referred to were not part of a shared visual scene as is the case in our data.

Gupta and Stent (2005) instantiated Dale and Reiter's (1995) Incremental Algorithm with a preference ordering that favours the attributes that were used in the previous mention of the same referent. In a second variant, they even require these attributes to be included in a subsequent reference. Differently from most other work on REG, they extended the task to include ordering of the attributes in the surface form. They therefore create a special evaluation metric that takes ordering into account, which makes it hard to compare the performance they report to that of any system that is not concerned with attribute ordering, such as ours. Their evaluation set was also considerably smaller than ours: they used 1294 and 471 referring expressions from two different corpora, compared to our test set of 4947 referring expressions.

More recently in (Viethen et al., 2010), we presented a rule-based system that addressed a specific instance of the problem we consider here, using the same corpus as we do: we singled out 2579 first references to landmarks by the second speaker ('second speaker initial references') and attempted to reproduce these using a system based on Dale and Reiter's (1995) Incremental Algorithm. Although the data set was a subset of the one used here, the system did not reach the same performance (see Section 5).

3 Referring Expressions in the iMAP Corpus

The iMAP Corpus (Louwerse et al., 2007) is a collection of 256 dialogues between 32 participant-pairs who contributed 8 dialogues each. Both participants had a map of the same environment, but one participant's map showed a route winding its way between the landmarks on the map; see Figure 1. The task was for this participant (the instruction giver, IG) to describe this route in such a way that their partner (the instruction follower, IF) could draw it onto their map; this was complicated by some discrepancies between the two maps, such

as missing landmarks, the unavailability of colour in some regions due to ink stains, and small differences between some landmarks.

The landmarks differ from each other in type, colour, and one other attribute, which is different for each type of landmark. For example, there are different *kinds* of birds (eagle, ostrich, penguin ...); fish differ by their *patterns* (dotted, checkered, plain ...), aliens have different *shapes* (circular, hexagonal ...), and bugs appear in small clusters of differing *numbers*. In addition to these inherent attributes of the landmarks, participants used spatial relations to other items on the map. Each referring expression in the corpus is annotated with a unique identifier corresponding to the landmark that it describes and the semantic values of the attributes that it contains. This collection of annotations forms the basic data we use in our experiments.

For each landmark R referred to in a dialogue, we view the sequence of references to this landmark as a *coreference chain*, notated $\langle R_1, R_2, \ldots, R_n \rangle$. By convention, R_1 is termed the *initial reference*, and all other references in the chain are *subsequent references*. From the corpus as a whole we extracted 34,127 referring expressions in 9558 chains. The average length of a chain is 4.74; and the longest coreference chain contains 43 references. References may be contributed to a chain by either speaker, and can be arbitrarily far apart: in the data, 4201 references are in the utterance immediately following the preceding reference in the chain, but the distance between references in a chain can be as high as 423 utterances.

We removed from the data any annotation that was not concerned with the four landmark attributes, type, colour, relation, or the landmark's other distinguishing attribute. For example, 'semantically empty' head nouns such as *thing* or *set*. Ordinal numbers that were annotated as the use of the number attribute were re-tagged as spatial relations, as these usually described the position of the target within a line of landmarks.

As a result of the removal of annotations not pertaining to the use of the four landmark attributes, 2785 referring expressions had no annotation left; we removed these instances from the final data set. We also do not attempt to replicate the remaining 5552 plural referring expressions or the 3062 pronouns found in the corpus.[1] However, we do include all of these instances in the feature extraction step, on the assumption that they might impact on the content of subsequent references. Similarly, we filter out 6369 initial references after we have extracted features from them, since we focus here on the generation of subsequent reference only. The remaining 16,358 referring expressions form the data which we use in our experiments.

Content Pattern	Count	Proportion
⟨other⟩	5893	36.0%
⟨other, type⟩	3684	22.5%
⟨other, colour⟩	1630	10.0%
⟨other, colour, type⟩	1021	6.2%
⟨colour⟩	969	5.9%
⟨relation⟩	777	4.7%
⟨other, relation⟩	587	3.6%
⟨type⟩	574	3.5%
⟨colour, type⟩	434	2.7%
⟨other, relation, type⟩	312	1.9%
⟨relation, type⟩	236	1.4%
⟨colour, relation⟩	99	0.6%
⟨other, colour, relation⟩	81	0.5%
⟨other, colour, relation, type⟩	44	0.3%
⟨colour, relation, type⟩	17	0.1%
Total	16,358	

Table 1: The 15 content patterns by frequency.

Contrary to findings from other corpora, in which colour was used much more frequently (Gatt, 2007; Viethen and Dale, 2008), the colour attribute was used in only 26.3% of the referring expressions in our data set. This is probably due to the often low reliability of colour in this task caused by the ink stains. The proportion of referring expressions mentioning the target's type might, at 38.7%, also seem low. This can be explained by the fact that one quarter of the landmarks, namely birds and buildings, are more likely to be described in terms of their specific kind than in terms of their generic type. This also helps explain why the overall use of the other attribute, which for some landmarks was their kind, was used in 81.0% of all instances. Spatial relations were used in 13.16% of the referring expressions, comparable to other corpora in the literature.

[1] The additional issues that arise in generating plural references and deciding when to use pronouns considerably complicate the problem; see (Gatt, 2007).

We can think of each referring expression as being a linguistic realisation of a *content pattern*: this is the collection of attributes that are used in that instance. The attributes can be derived from the property-level annotation given in the corpus. So, for example, if a particular reference appears as the noun phrase *the blue penguin*, annotated semantically as $\langle \mathsf{blue}, \mathsf{penguin} \rangle$, then the corresponding content pattern is $\langle \mathsf{colour}, \mathsf{kind} \rangle$. Our aim is to replicate the content pattern of each referring expression in the corpus. Table 1 lists the 15 content patterns that occur in our data set in order of frequency.

4 Modelling Referential Behaviour

4.1 The Two Perspectives

Our task is defined simply as follows: for each subsequent reference R in the corpus, can we predict the content pattern that will be used in that reference? As we noted at the outset of the paper, the literature would appear to suggest two distinct approaches to this problem. What we have characterised as the algorithmic approach can be summarised thus:

> At the point where a reference is required, a speaker determines the relevant features of other entities in the context, then computes the content of a referring expression which distinguishes the intended referent from the other entities.

The alignment approach, on the other hand, can be summarised thus:

> Speakers align the forms of reference they use to be similar or identical to references that have been used before. In particular, once a form of reference to the intended referent has been established, they tend to re-use that form of reference, or perhaps an abbreviated version of it.

The alignment approach would appear to be preferable on the grounds of computational cost: we would expect that retrieving a previously-used referring expression, or parts thereof, generally requires less computation than building a new referring expression from scratch.

On the other hand, if the context has changed in any way, then a previously-used form of reference may no longer be effective in identifying the intended referent, and recomputation may be required.[2] This is precisely the consideration on which the initial work on referring expression generation was based, inspired by Grosz and Sidner's (1986) observations about how the changing attentional structure of a discourse moves different entities in and out of focus. However, a straightforward recomputation of reference based on the current context carries the risk that the most effective set of properties to use may change quite radically; if no account is taken of the history of previous references to the entity, it's conceivable that one could produce a description that is so different from the previous description that they are virtually unrecogisable as descriptions of the same entity. Ideally, what we want to do is *modify* a previous description to do the job.

These observations suggest that, in order to choose the most appropriate form of reference for an entity, we need to simultaneously take account of:

- the other entities from which it must be distinguished, both in the visual context and in the preceding discourse (in other words, exactly the information that traditional algorithmic approaches consider);
- how this entity, and perhaps other entities, have been referred to in the past (precisely the information that the alignment approach considers).

Map Features	
Main_Map_type	most frequent type of LM on this map
Main_Map_other	other attribute if the most frequent type of LM
Mixedness	are other LM types present on this map?
Ink_Orderliness	shape of the ink blot(s) on the IF's map
Lmprop Features	
other_Att	type of the other attribute of the target
*[att]*_Value	value for each *att* of target
*[att]*_Difference	was *att* of target different between the two maps?
Missing	was target missing one of the maps?
Inked_Out	was target inked]_out on the IG's map?
Speaker Features	
Dyad_ID	ID of the pair of participant-pair
Speaker_ID	ID of the person who uttered this RE
Speaker_Role	was the speaker the IG or the IF?

Table 2: The Ind feature set.

[2] Unfortunately, determining what counts as a change of context, especially in visual scenes, is fraught with difficulty.

TradREG Features (Visual)	
Count_Vis_Distractors	number of visual distractors
Prop_Vis_Same_*[att]*	proportion of visual distractors with same *att*
Dist_Closest	distance to the closest visual distractor
Closest_Same_*[att]*	has the closest distractor the same *att*?
Dist_Closest_Same_*[att]*	distance to the closest distractor of same *att* as target
Cl_Same_type_Same_*[att]*	has the closest distractor of the same type also the same *att*?
TradREG Features (Discourse)	
Count_Intervening_LMs	number of other LMs mentioned since the last mention of the target
Prop_Intervening_*[att]*	proportion of intervening LMs for which *att* was used AND which have the same *att* as target

Table 3: The TradREG feature set.

Alignment Features (Recency)	
Last_Men_Speaker_Same	who made the last mention of target?
Last_Mention_*[att]*	was *att* used in the last mention of target?
Dist_Last_Mention_Utts	distance to the last mention of target in utterances
Dist_Last_Mention_REs	distance to the last mention of target in REs
Dist_Last_*[att]*_LM_Utts	distance in utterances to last use of *att* for target
Dist_Last_*[att]*_LM_REs	distance in REs to last use of *att* for target
Dist_Last_*[att]*_Dial_Utts	distance in utterances to last use of *att*
Dist_Last_*[att]*_Dial_REs	distance in REs to last use of *att*
Dist_Last_RE_Utts	distance to last RE in utterances
Last_RE_*[att]*	was *att* mentioned in the last RE?
Alignment Features (Frequency)	
Count_*[att]*_Dial	how often has *att* been used in the dialogue?
Count_*[att]*_LM	how often has *att* been used for target?
Quartile	quartile of the dialogue the RE was uttered in
Dial_No	number of dialogues already completed +1
Mention_No	number of previous mentions of target +1

Table 4: The Alignment feature set.

The set of features we describe next attempts to capture these two aspects of the problem.

4.2 Features

The number of factors that can be hypothesised as having an impact on the form of a referring expression in a dialogic setting associated with a visual domain is very large. Attempting to incorporate all of these factors into parameters for rule-based systems, and then experimenting with different settings for these parameters, is prohibitively complex. Instead, we here capture a wide range of factors as features that can be used by a machine learning algorithm to automatically induce from the data a classifier that predicts for a given set of features the attributes that should be used in a referring expression.

The features we extracted from the data set are listed in Tables 2–4.[3] They fall into five subsets. **Map** Features capture design characteristics of the maps the current dialogue is about; **Speaker** Features capture the identity and role of the participants; and **LMprop** Features capture the inherent visual properties of the target referent. For our experiments, we group the Map, LMprop and Speaker feature sets into one theory-independent set (**Ind**). Most importantly for our present considerations, **TradREG** Features capture factors that the traditional computational approaches to referring expression generation take account of, in particular properties of the discourse and visual distractors; and **Alignment** Features capture factors that we would expect to play a role in the psycholinguistic models of alignment and conceptual pacts.

[3] In these tables, *att* is an abbreviatory variable that is instantiated once for each of the four attributes type, colour, relation, and the other distinguishing attribute of the landmark. The abbreviation LM stands for landmark

4.3 The Models

For the experiments described here, we used a 70–30 split to divide the data into a training set (11,411 instances) and a test set (4,947 instances). In addition to the main prediction class *content pattern*, the split was stratified for Speaker_ID and Quartile to ensure that training and test set contained the same proportion of descriptions from each speaker and each quartile of the dialogues. We used the J48 algorithm implemented in the Weka toolkit (Witten and Frank, 2005) to train decision trees with the task of judging, based on the given features, which content pattern should be used.

First, we have three separate baseline models:

HeadNounOnly generates only the property that is the most likely head noun for the target, which is kind for birds and buildings and type for all

other landmarks. This is a form of 'reduced reference' strategy.

RepeatLast represents a very simplistic alignment approach. It generates the same content pattern that was used in the previous mention of the target referent.

MajorityClass generates the content pattern most commonly used in the training set.

We then have a number of models that use subsets of the features described above:

AllFeatures is a decision tree trained on all features;

TradREG is a decision tree trained on the TradREG features only;

Alignment is a decision tree trained on the Alignment features only;

Ind is a decision tree trained on the Ind features only;

Alignment+Ind is a decision tree trained on all but the TradREG features;

TradREG+Ind is a decision tree trained on all but the Alignment features; and

TradREG+Alignment is a decision tree trained on all but the Ind features.

5 Results

In this section we report how the models described in the previous section performed on the held-out test set in comparison to each other and to the three baselines.

We use Accuracy and average DICE score for our comparisons; these are the most commonly used measures in the REG literature (see, for example, Gatt et al., 2008). Given two sets of attributes, A and B, DICE is computed as

$$\text{DICE} = \frac{2 \times |A \cap B|}{|A| + |B|}. \quad (1)$$

This gives some measure of the overlap between two referring expressions, assigning a partial score if the two sets share attributes but are not identical. The Accuracy of a system is the proportion of test instances for which it achieves a DICE score of 1, signifying a perfect match.

	col	other	type	rel	Comb. Pattern	
	Acc	Acc	Acc	Acc	Acc	DICE
HeadOnly	n/a	n/a	n/a	n/a	23.1	0.49
RepLast	n/a	n/a	n/a	n/a	38.4	0.55
Majority	73.8	81.0	61.7	86.8	36.0	0.65
predicts	no	yes	no	no	⟨other⟩	
Trad	74.6	84.8	77.1	87.0	47.3	0.73
Align	83.6	84.1	80.7	87.5	54.6	0.78
Ind	81.9	82.8	81.4	88.0	52.7	0.78
Align+Ind	86.1	85.3	82.4	**88.7**	58.2	**0.81**
Trad+Ind	82.2	84.1	81.1	87.1	52.5	0.78
Trad+Align	84.1	84.0	80.1	86.8	53.9	0.78
AllFeatures	**86.2**	**85.8**	**83.2**	88.5	**58.8**	**0.81**

Table 5: Performance of our models compared to the baselines. Model names are abbreviated for space reasons. The Accuracy (given in %) of all models is significantly better than that of the highest performing baseline at p<.01 according to the χ^2 statistic.

We tested two different ways of generating content patterns based on the different feature sets described above: **PatternAtOnce** builds a decision tree that chooses one of the 15 content patterns that occur in our data set; whereas **CombinedPattern** builds attribute-specific decision trees (one for each of the four attributes that occur in the data: colour, other, type, and relation), and then combines their predictions into a complete content pattern. We found that CombinedPattern slightly outperformed PatternAtOnce, although the difference is not statistically significant for all feature sets. For space reasons, we report in what follows only on the slightly better-performing CombinedPattern model.

Table 5 compares the performances of the three baselines and the decision trees based on the five feature subsets for each of the individual attributes and for the combined content pattern; note that the HeadNounOnly and RepeatLast baselines do not make attribute-specific predictions. The table shows that the learned systems outperform all three baselines for the individual attributes as well as for the combined content pattern.

A comparison of the Alignment feature set and the TradREG feature set shows that the former outperforms the latter for the attribute-specific trees which predict the use of the colour attribute and the

use of relation, and that the combined patterns resulting from the Alignment trees are a better match of the human-produced patterns both in terms of Accuracy ($p<.01$ for all three categories, using χ^2) and DICE. Interestingly, even the theory-independent Ind features outperform the TradREG features.

The comparison between TradREG+Ind and Alignment+Ind again shows a clear advantage for the Alignment features: dropping them from the complete feature set significantly hurts performance compared to AllFeatures (χ^2=80.5, $p<.01$), while dropping the TradREG features has no significant impact. Also consistent with the results of the three individual feature sets, dropping the Ind features hurts performance more than dropping the TradREG features, but less than dropping the Alignment features. Training on the complete feature set (AllFeatures) achieves the highest performance, which is significantly better than that of all other features sets ($p<.01$ using χ^2) except Alignment+Ind.

These results suggest that considerations at the heart of traditional REG approaches do not play as important a role as those postulated by alignment-based models for the selection of semantic content for subsequent referring expressions.

We also note that the Accuracy scores achieved by our learned systems are similar to the best numbers previously reported in the REG literature. While Jordan and Walker's (2005) data set is not directly comparable, they achieved a maximum of 59.9% Accuracy, against our 58.8%. Stoia et al.'s (2006) best Accuracy was 31.2%, albeit on a slightly different task. Even in the arguably much simpler non-dialogic domains of the REG competitions concerned with pure content selection, the best performing system achieved only 53% Accuracy (see Gatt et al., 2008). The most comparable approach, the rule-based system we presented in (Viethen et al., 2010) for a subset of the data used here, was not able to outperform a RepeatLast baseline at 40.2% Accuracy and an average DICE score of 0.67.

6 Error Analysis

An important question to ask is how wrong the models really are when they do not succeed in perfectly matching a human-produced reference in our test set. It might be that they choose a completely different set of attributes from those included by the human speaker; however, the Accuracy score also counts as incorrect any set that only partly overlaps with the reference found in the test set.

	Acc	Dice	Super	Sub	Inter	Noover
Trad	47.3	0.75	14.4	22.2	5.5	10.5
Align	54.6	0.78	16.0	16.1	3.9	9.4
Ind	52.7	0.78	17.1	17.2	3.9	9.0
Align+Ind	58.2	0.81	16.0	14.8	3.1	7.9
Trad+Ind	52.5	0.78	17.4	17.5	3.8	8.8
Trad+Align	53.9	0.78	17.1	15.6	4.3	9.0
AllFeature	58.8	0.81	16.5	14.5	3.1	7.2

Table 6: The proportions of test instances for which each model produced a subset, a superset, some other form of intersection or no-overlap to the human reference.

The DICE score gives us a partial answer to this question, as it assigns a score that is based on the size of the overlap between the attribute set chosen by the model and that included by the human speaker. A DICE score that is equal to the Accuracy score would mean that each referring expression was either reproduced perfectly, or that a set of attributes was chosen that did not overlap with the original one at all. The fact that all our models achieved DICE scores much higher than their Accuracy scores shows that they only rarely got it completely wrong.

Table 6 gives a more fine-grained picture by listing, for each model, what percentage of the referring expressions it produced contained a subset of the attributes included in the human reference, what percentage were a superset, what percentage had another form of partial intersection, and what percentage had no commonality with the human reference. Interestingly, a large number of the referring expressions produced by the model trained only on TradREG features are subsets of the human reference. This indicates that human speakers tend to include more attributes than are strictly speaking necessary to distinguish the landmark.[4] The Alignment model does not as often produce a subset of the gold standard content pattern, suggesting that it might be alignment considerations that account for some of

[4]That humans often produce 'redundant' descriptions, in opposition to the target behaviour of some of the early REG algorithms, is of course an oft-observed fact.

	both corr.	both wrong	1st corr.	2nd corr.	either corr.	pot. Acc
Trad vs Ind	1797	1794	545	811	3153	63.7
Trad vs Align	1742	1647	600	958	3300	66.7
Trad vs Align+Ind	1849	1574	493	1031	3373	68.2
Align vs Trad+Ind	1908	1557	792	690	3390	68.5
Align vs Ind	1872	1511	828	736	3436	69.5
Ind vs Trad+Align	1840	1511	768	828	3436	69.5

Table 7: Comparison of the predictions for the combined content pattern between the models trained on mutually exclusive feature sets.

the apparent redundancy that human-produced referring expressions contain.

A second important question is whether the different feature sets are doing the same work, or whether they complement each other. Table 7 lists for those pairings of our learned models which were based on mutually exclusive feature sets how many referring expressions both models predicted correctly, how many both failed to predict, and how many were predicted correctly by either of the two models.

Note the high numbers in the columns listing the counts of instances which both models got either correct or wrong: these show that there is considerable overlap between all pairings. The smallest agreement lies at 3424 instances (68.2%) between TradREG (the least successful model) and Alignment+Ind (the most successful model). However, they also each predict correct solutions that the other misses: 493 (10.0%) for TradREG and 1031 (20.8%) for Alignment+Ind.

The last two columns of Table 7 show the number of instances that at least one of the two models in each pairing got correct and the proportion out of all test instances that this number represents. This proportion is the maximum Accuracy that could be achieved by a model that combines the two models in a pairing and then correctly chooses which one to use in each instance. The maximum Accuracies that could be achieved in this way on our data set lie just below 70%, significantly higher than any numbers reported in the literature on the task of generating subsequent reference.

7 Conclusions

Using the largest corpus of referring expressions to date, we have shown how both the traditional computational view of REG and the alternative psycholinguistic alignment approach can be captured via a large set of features for machine learning. Additionally, we defined a number of theory independent features. Using this approach we have presented three main findings.

First, we have demonstrated that a model using all these features to predict content patterns in subsequent references in shared visual scenes delivers an Accuracy of 58.8% and a DICE score of 0.81, outperforming models based only on features inspired by one of the two approaches. However, we found that the features based on traditional REG considerations do not contribute as much to this score as those based on the alignment approach, and that dropping the traditional REG features does not significantly hurt the performance of a model based on alignment and theory-independent features.

Second, our error analysis showed that the main reason for the low performance of a model based on traditional algorithmic features is that it often chooses too few attributes. The fact that the model based on the alignment features does not make this mistake so frequently suggests that it may be the psycholinguistic considerations incorporated in our alignment features that lead people to add those additional attributes.

Finally, while the different models make the same correct predictions about the content of referring expressions in many cases, there are also a considerable number of cases where the models based on either the traditional algorithmic features (10.0%) or the alignment and independent features (20.8%) alone make correct predictions that the other gets wrong; this suggests that a system with the ability to choose the correct model in each of those cases (perhaps based on a hypothesis as to whether or not the relevant context has changed) could reach an accuracy of almost 70% on our data set. In future work we plan to identify further features that will allow us to inform this choice so that we can move towards this level of performance.

References

Susan E. Brennan and Herbert H. Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22:1482–1493.

Hendrik Buschmeier, Kirsten Bergmann, and Stefan Kopp. 2009. An alignment-capable microplanner for natural language generation. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 82–89, Athens, Greece.

John M. Carroll. 1980. Naming and describing in social communication. *Language and Speech*, 23:309–322.

Herbert H. Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22(1):1–39.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

Robert Dale. 1989. Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver B.C., Canada.

Claire Gardent and Kristina Striegnitz. 2007. Generating bridging definite descriptions. In Harry C. Bunt and Reinhard Muskens, editors, *Computing Meaning*, volume 3, pages 369–396. Kluwer, Dordrecht, The Netherlands.

Albert Gatt, Anja Belz, and Eric Kow. 2008. The TUNA Challenge 2008: Overview and evaluation results. In *Proceedings of the 5th International Conference on Natural Language Generation*, pages 198–206, Salt Fork OH, USA.

Albert Gatt. 2007. *Generating Coherent Reference to Multiple Entities*. Ph.D. thesis, University of Aberdeen, UK.

Martijn Goudbeek and Emiel Krahmer. 2010. Preferences versus adaptation during referring expression generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 55–59, Uppsala, Sweden.

Barbara J. Grosz and Candance L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Surabhi Gupta and Amanda Stent. 2005. Automatic evaluation of referring expression generation using corpora. In *Proceedings of the Workshop on Using Corpora for Natural Language Generation*, pages 1–6, Brighton, UK.

Srinivasan Janarthanam and Oliver Lemon. 2009. Learning lexical alignment policies for generating referring expressions for spoken dialogue systems. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 74–81, Athens, Greece, March. Association for Computational Linguistics.

Pamela W. Jordan and Marilyn Walker. 2000. Learning attribute selections for non-pronominal expressions. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, Hong Kong, China.

Pamela W. Jordan and Marilyn Walker. 2005. Learning content selection rules for generating object descriptions in dialogue. *Journal of Artificial Intelligence Research*, 24:157–194.

Pamela W. Jordan. 2000. *Intentional Influences on Object Redescriptions in Dialogue: Evidence from an Empirical Study*. Ph.D. thesis, University of Pittsburgh, Pittsburgh PA, USA.

Emiel Krahmer, Sebastiaan van Erk, and André Verleg. 2003. Graph-based generation of referring expressions. *Computational Lingustics*, 29(1):53–72.

Max M. Louwerse, Nick Benesh, Mohammed E. Hoque, Patrick Jeuniaux, Gwyneth Lewis, Jie Wu, and Megan Zirnstein. 2007. Multimodal communication in face-to-face computer-mediated conversations. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, pages 1235–1240.

Martin J. Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27(2):169–226.

Laura Stoia, Darla M. Shockley, Donna K. Byron, and Eric Fosler-Lussier. 2006. Noun phrase generation for situated dialogs. In *Proceedings of the 4th International Conference on Natural Language Generation*, pages 81–88, Sydney, Australia, July.

Kees van Deemter and Emiel Krahmer. 2007. Graphs and Booleans: On the generation of referring expressions. In Harry C. Bunt and Reinhard Muskens, editors, *Computing Meaning*, volume 3, pages 397–422. Kluwer, Dordrecht, The Netherlands.

Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In *Proceedings of the 5th International Conference on Natural Language Generation*, pages 59–67, Salt Fork OH, USA.

Jette Viethen, Simon Zwarts, Robert Dale, and Markus Guhe. 2010. Dialogue reference in a visual domain. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, Valetta, Malta.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco CA, USA.

Learning Sentential Paraphrases from Bilingual Parallel Corpora for Text-to-Text Generation

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and **Benjamin Van Durme**
Center for Language and Speech Processing, and HLTCOE
Johns Hopkins University

Abstract

Previous work has shown that high quality *phrasal* paraphrases can be extracted from bilingual parallel corpora. However, it is not clear whether bitexts are an appropriate resource for extracting more sophisticated *sentential* paraphrases, which are more obviously learnable from monolingual parallel corpora. We extend bilingual paraphrase extraction to syntactic paraphrases and demonstrate its ability to learn a variety of general paraphrastic transformations, including passivization, dative shift, and topicalization. We discuss how our model can be adapted to many text generation tasks by augmenting its feature set, development data, and parameter estimation routine. We illustrate this adaptation by using our paraphrase model for the task of sentence compression and achieve results competitive with state-of-the-art compression systems.

1 Introduction

Paraphrases are alternative ways of expressing the same information (Culicover, 1968). Automatically generating and detecting paraphrases is a crucial aspect of many NLP tasks. In multi-document summarization, paraphrase detection is used to collapse redundancies (Barzilay et al., 1999; Barzilay, 2003). Paraphrase generation can be used for query expansion in information retrieval and question answering systems (McKeown, 1979; Anick and Tipirneni, 1999; Ravichandran and Hovy, 2002; Riezler et al., 2007). Paraphrases allow for more flexible matching of system output against human references for tasks like machine translation and automatic summarization (Zhou et al., 2006; Kauchak and Barzilay, 2006; Madnani et al., 2007; Snover et al., 2010).

Broadly, we can distinguish two forms of paraphrases: *phrasal paraphrases* denote a set of surface text forms with the same meaning:

the committee's second proposal
the second proposal of the committee

while *syntactic paraphrases* augment the surface forms by introducing nonterminals (or *slots*) that are annotated with syntactic constraints:

the NP_1's NP_2
the NP_2 of the NP_1

It is evident that the latter have a much higher potential for generalization and for capturing interesting paraphrastic transformations.

A variety of different types of corpora (and semantic equivalence cues) have been used to automatically induce paraphrase collections for English (Madnani and Dorr, 2010). Perhaps the most natural type of corpus for this task is a monolingual parallel text, which allows sentential paraphrases to be extracted since the sentence pairs in such corpora are perfect paraphrases of each other (Barzilay and McKeown, 2001; Pang et al., 2003). While rich syntactic paraphrases have been learned from monolingual parallel corpora, they suffer from very limited data availability and thus have poor coverage.

Other methods obtain paraphrases from raw monolingual text by relying on distributional similarity (Lin and Pantel, 2001; Bhagat and Ravichandran, 2008). While vast amounts of data are readily available for these approaches, the distributional similarity signal they use is noisier than the sentence-level correspondency in parallel corpora and additionally suffers from problems such as mistaking cousin expressions or antonyms (such as $\{boy, girl\}$ or $\{rise, fall\}$) for paraphrases.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1168–1179,
Edinburgh, Scotland, UK, July 27–31, 2011.

Abundantly available bilingual parallel corpora have been shown to address both these issues, obtaining paraphrases via a pivoting step over foreign language phrases (Bannard and Callison-Burch, 2005). The coverage of paraphrase lexica extracted from bitexts has been shown to outperform that obtained from other sources (Zhao et al., 2008a). While there have been efforts pursuing the extraction of more powerful paraphrases (Madnani et al., 2007; Callison-Burch, 2008; Cohn and Lapata, 2008; Zhao et al., 2008b), it is not yet clear to what extent sentential paraphrases can be induced from bitexts. In this paper we:

- Extend the bilingual pivoting approach to paraphrase induction to produce rich syntactic paraphrases.
- Perform a thorough analysis of the types of paraphrases we obtain and discuss the paraphrastic transformations we are capable of capturing.
- Describe how training paradigms for syntactic/sentential paraphrase models should be tailored to different text-to-text generation tasks.
- Demonstrate our framework's suitability for a variety of text-to-text generation tasks by obtaining state-of-the-art results on the example task of sentence compression.

2 Related Work

Madnani and Dorr (2010) survey a variety of data-driven paraphrasing techniques, categorizing them based on the type of data that they use. These include large monolingual texts (Lin and Pantel, 2001; Szpektor et al., 2004; Bhagat and Ravichandran, 2008), comparable corpora (Barzilay and Lee, 2003; Dolan et al., 2004), monolingual parallel corpora (Barzilay and McKeown, 2001; Pang et al., 2003), and bilingual parallel corpora (Bannard and Callison-Burch, 2005; Madnani et al., 2007; Zhao et al., 2008b). We focus on the latter type of data.

Paraphrase extraction using bilingual parallel corpora was proposed by Bannard and Callison-Burch (2005) who induced paraphrases using techniques from *phrase-based* statistical machine translation (Koehn et al., 2003). After extracting a bilingual phrase table, English paraphrases are obtained by pivoting through foreign language phrases. Since many paraphrases can be extracted for a phrase, Bannard and Callison-Burch rank them using a paraphrase probability defined in terms of the translation model probabilities $p(f|e)$ and $p(e|f)$:

$$p(e_2|e_1) = \sum_f p(e_2, f|e_1) \quad (1)$$

$$= \sum_f p(e_2|f, e_1)p(f|e_1) \quad (2)$$

$$\approx \sum_f p(e_2|f)p(f|e_1). \quad (3)$$

Several subsequent efforts extended the bilingual pivoting technique, many of which introduced elements of more contemporary *syntax-based* approaches to statistical machine translation. Madnani et al. (2007) extended the technique to *hierarchical* phrase-based machine translation (Chiang, 2005), which is formally a synchronous context-free grammar (SCFG) and thus can be thought of as a *paraphrase grammar*. The paraphrase grammar can paraphrase (or "decode") input sentences using an SCFG decoder, like the Hiero, Joshua or cdec MT systems (Chiang, 2007; Li et al., 2009; Dyer et al., 2010). Like Hiero, Madnani's model uses just one nonterminal X instead of linguistic nonterminals.

Three additional efforts incorporated linguistic syntax. Callison-Burch (2008) introduced syntactic constraints by labeling all phrases and paraphrases (even non-constituent phrases) with CCG-inspired slash categories (Steedman and Baldridge, 2011), an approach similar to Zollmann and Venugopal (2006)'s syntax-augmented machine translation (SAMT). Callison-Burch did not formally define a synchronous grammar, nor discuss decoding, since his presentation did not include hierarchical rules. Cohn and Lapata (2008) used the GHKM extraction method (Galley et al., 2004), which is limited to constituent phrases and thus produces a reasonably small set of syntactic rules. Zhao et al. (2008b) added slots to bilingually extracted paraphrase patterns that were labeled with part-of-speech tags, but not larger syntactic constituents.

Before the shift to statistical natural language processing, paraphrasing was often treated as syntactic transformations or by parsing and then generating

from a semantic representation (McKeown, 1979; Muraki, 1982; Meteer and Shaked, 1988; Shemtov, 1996; Yamamoto, 2002). Indeed, some work generated paraphrases using (non-probabilistic) synchronous grammars (Shieber and Schabes, 1990; Dras, 1997; Dras, 1999; Kozlowski et al., 2003).

After the rise of statistical machine translation, a number of its techniques were repurposed for paraphrasing. These include sentence alignment (Gale and Church, 1993; Barzilay and Elhadad, 2003), word alignment and noisy channel decoding (Brown et al., 1990; Quirk et al., 2004), phrase-based models (Koehn et al., 2003; Bannard and Callison-Burch, 2005), hierarchical phrase-based models (Chiang, 2005; Madnani et al., 2007), log-linear models and minimum error rate training (Och, 2003a; Madnani et al., 2007; Zhao et al., 2008a), and here syntax-based machine translation (Wu, 1997; Yamada and Knight, 2001; Melamed, 2004; Quirk et al., 2005).

Beyond cementing the ties between paraphrasing and syntax-based statistical machine translation, the novel contributions of our paper are (1) an in-depth analysis of the types of structural and sentential paraphrases that can be extracted with bilingual pivoting, (2) a discussion of how our English–English paraphrase grammar should be adapted to specific text-to-text generation tasks (Zhao et al., 2009) with (3) a concrete example of the adaptation procedure for the task of paraphrase-based sentence compression (Knight and Marcu, 2002; Cohn and Lapata, 2008; Cohn and Lapata, 2009).

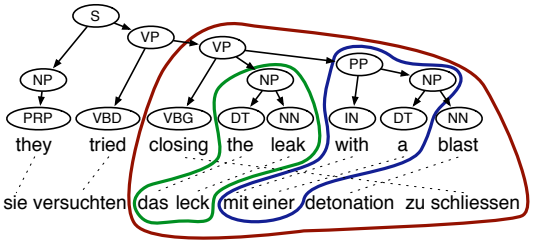
3 SCFGs in Translation

The model we use in our paraphrasing approach is a syntactically informed *synchronous context-free grammar* (SCFG). The SCFG formalism (Aho and Ullman, 1972) was repopularized for statistical machine translation by Chiang (2005). Formally, a *probabilistic* SCFG $\mathcal{G}$ is defined by specifying

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{T}_S, \mathcal{T}_T, \mathcal{R}, S \rangle,$$

where $\mathcal{N}$ is a set of nonterminal symbols, $\mathcal{T}_S$ and $\mathcal{T}_T$ are the source and target language vocabularies, $\mathcal{R}$ is a set of rules and $S \in \mathcal{N}$ is the root symbol. The rules in $\mathcal{R}$ take the form:

$$C \rightarrow \langle \gamma, \alpha, \sim, w \rangle,$$



NP → das leck | the leak
PP/NN → mit einer | with a
VP → NP PP/NN detonation zu schliessen | closing NP PP/NN blast

Figure 1: Synchronous grammar rules for translation are extracted from sentence pairs in a bixtext which have been automatically parsed and word-aligned. Extraction methods vary on whether they extract only minimal rules for phrases dominated by nodes in the parse tree, or more complex rules that include non-constituent phrases.

where the rule's left-hand side $C \in \mathcal{N}$ is a nonterminal, $\gamma \in (\mathcal{N} \cup \mathcal{T}_S)^*$ and $\alpha \in (\mathcal{N} \cup \mathcal{T}_T)^*$ are strings of terminal and nonterminal symbols with an equal number of nonterminals $c_{NT}(\gamma) = c_{NT}(\alpha)$ and

$$\sim: \{1 \ldots c_{NT}(\gamma)\} \rightarrow \{1 \ldots c_{NT}(\alpha)\}$$

constitutes a one-to-one correspondency function between the nonterminals in γ and α. A non-negative weight $w \geq 0$ is assigned to each rule, reflecting the likelihood of the rule.

Rule Extraction Phrase-based approaches to statistical machine translation (and their successors) extract pairs of (e, f) phrases from automatically word-aligned parallel sentences. Och (2003b) described various heuristics for extracting phrase alignments from the Viterbi word-level alignments that are estimated using Brown et al. (1993) word-alignment models.

These phrase extraction heuristics have been extended so that they extract synchronous grammar rules (Galley et al., 2004; Chiang, 2005; Zollmann and Venugopal, 2006; Liu et al., 2006). Most of these extraction methods require that one side of the parallel corpus be parsed. This is typically done automatically with a statistical parser.

Figure 1 shows examples of rules obtained from a sentence pair. To extract a rule, we first choose a source side span f like *das leck*. Then we use phrase extraction techniques to find target spans e that are consistent with the word alignment (in this case *the*

leak is consistent with our f). The nonterminal symbol that is the left-hand side of the SCFG rule is then determined by the syntactic constituent that dominates e (in this case *NP*). To introduce nonterminals into the right-hand side of the rule, we can apply rules extracted over sub-phrases of f, synchronously substituting the corresponding nonterminal symbol for the sub-phrases on both sides. The synchronous substitution applied to f and e then yields the correspondency $\sim$.

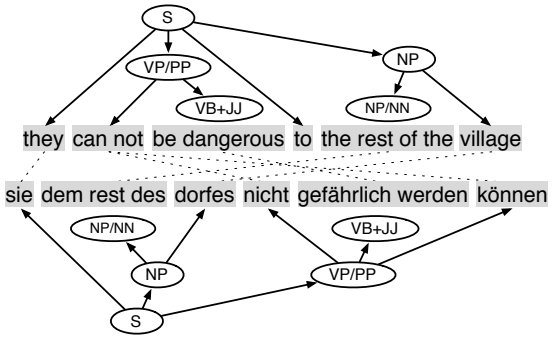
One significant differentiating factor between the competing ways of extracting SCFG rules is whether the extraction method generates rules only for constituent phrases that are dominated by a node in the parse tree (Galley et al., 2004; Cohn and Lapata, 2008) or whether they include arbitrary phrases, including non-constituent phrases (Zollmann and Venugopal, 2006; Callison-Burch, 2008). We adopt the extraction for all phrases, including non-constituents, since it allows us to cover a much greater set of phrases, both in translation and paraphrasing.

Feature Functions Rather than assigning a single weight w, we define a set of feature functions $\vec{\varphi} = \{\varphi_1 ... \varphi_N\}$ that are combined in a log-linear model:

$$w = -\sum_{i=1}^{N} \lambda_i \log \varphi_i. \quad (4)$$

The weights $\vec{\lambda}$ of these feature functions are set to maximize some objective function like BLEU (Papineni et al., 2002) using a procedure called minimum error rate training (MERT), owing to Och (2003a). MERT iteratively adjusts the weights until the decoder produces output that best matches reference translations in a development set, according to the objective function. We will examine appropriate objective functions for text-to-text generation tasks in Section 6.2.

Typical features used in statistical machine translation include phrase translation probabilities (calculated using maximum likelihood estimation over all phrase pairs enumerable in the parallel corpus), word-for-word lexical translation probabilities (which help to smooth sparser phrase translation estimates), a "rule application penalty" (which governs whether the system prefers fewer longer phrases or a greater number of shorter phrases), and a language model probability.



NP/NN → dem rest des | the rest of the
VB+JJ → gefährlich werden | be dangerous
NP → NP/NN dorfes | NP/NN village
VP/PP → nicht VB+JJ können | can not VB+JJ
S → sie NP VP/PP | they VP/PP to NP

Figure 2: An example derivation produced by a syntactic machine translation system. Although the synchronous trees are unlike the derivations found in the Penn Treebank, their yield is a good translation of the German.

Decoding Given an SCFG and an input source sentence, the decoder performs a search for the single most probable derivation via the CKY algorithm. In principle the best translation should be the English sentence e that is the most probable after summing over all $d \in D$ derivations, since many derivations yield the same e. In practice, we use a Viterbi approximation and return the translation that is the yield of the single best derivation:

$$\begin{aligned} \hat{e} &= \arg \max_{e \in Trans(f)} \sum_{d \in D(e,f)} p(d, e|f) \\ &\approx yield(\arg \max_{d \in D(e,f)} p(d, e|f)). \end{aligned} \quad (5)$$

Derivations are simply successive applications of the SCFG rules such as those given in Figure 2.

4 SCFGs in Paraphrasing

Rule Extraction To create a paraphrase grammar from a translation grammar, we extend the syntactically informed pivot approach of Callison-Burch (2008) to the SCFG model. For this purpose, we assume a grammar that translates from a given foreign language to English. For each pair of translation rules where the left-hand side C and foreign

string γ match:

$$C \rightarrow \langle \gamma, \alpha_1, \sim_1, \vec{\varphi}_1 \rangle$$
$$C \rightarrow \langle \gamma, \alpha_2, \sim_2, \vec{\varphi}_2 \rangle,$$

we create a paraphrase rule:

$$C \rightarrow \langle \alpha_1, \alpha_2, \sim, \vec{\varphi} \rangle,$$

where the nonterminal correspondency relation $\sim$ has been set to reflect the combined nonterminal alignment:

$$\sim \;\; = \;\; \sim_1^{-1} \circ \sim_2 \; .$$

Feature Functions In the computation of the features $\vec{\varphi}$ from $\vec{\varphi}_1$ and $\vec{\varphi}_2$ we follow the approximation in Equation 3, which yields lexical and phrasal paraphrase probability features. Additionally, we add a boolean indicator for whether the rule is an identity paraphrase, $\delta_{identity}$. Another indicator feature, $\delta_{reorder}$, fires if the rule swaps the order of two nonterminals, which enables us to promote more complex paraphrases that require structural reordering.

Decoding With this, paraphrasing becomes an English-to-English translation problem which can be formulated similarly to Equation 5 as:

$$\hat{e_2} \approx yield(\arg \max_{d \in D(e_2, e_1)} p(d, e_2 | e_1)).$$

Figure 3 shows an example derivation produced as a result of applying our paraphrase rules in the decoding process. Another advantage of using the decoder from statistical machine translation is that n-gram language models, which have been shown to be useful in natural language generation (Langkilde and Knight, 1998), are already well integrated (Huang and Chiang, 2007).

5 Analysis

A key motivation for the use of syntactic paraphrases over their phrasal counterparts is their potential to capture meaning-preserving linguistic transformations in a more general fashion. A phrasal system is limited to memorizing fully lexicalized transformations in its paraphrase table, resulting in poor generalization capabilities. By contrast, a syntactic paraphrasing system intuitively should be able to address this issue and learn well-formed and generic patterns that can be easily applied to unseen data.

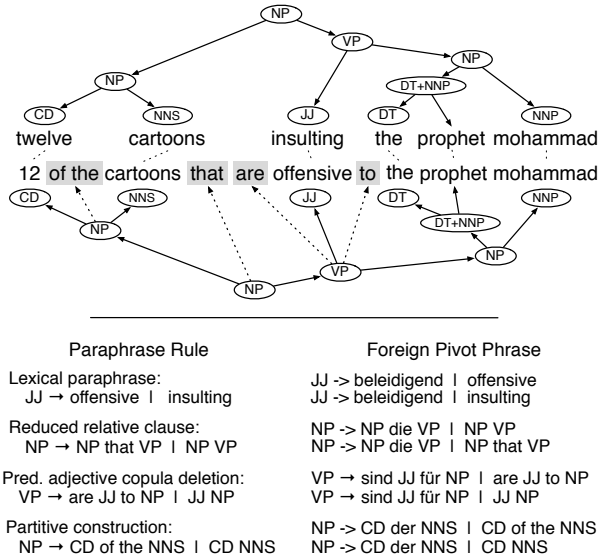


Figure 3: An example of a synchronous paraphrastic derivation. A few of the rules applied in the parse are show in the left column, with the pivot phrases that gave rise to them on the right.

To put this expectation to the test, we investigate how our grammar captures a number of well-known paraphrastic transformations.[1] Table 1 shows the transformations along with examples of the generic grammar rules our system learns to represent them. When given a transformation to extract a syntactic paraphrase for, we want to find rules that neither under- nor over-generalize. This means that, while replacing the maximum number of syntactic arguments with nonterminals, the rules ideally will both retain enough lexicalization to serve as sufficient evidence for the applicability of the transformation and impose constraints on the nonterminals to ensure the arguments' well-formedness.

The paraphrases implementing the *possessive rule* and the *dative shift* shown in Table 1 are a good examples of this: the two noun-phrase arguments to the expressions are abstracted to nonterminals while each rule's lexicalization provides an appropriate frame of evidence for the transform. This is important for a good representation of dative shift, which is a reordering transformation that fully applies to certain ditransitive verbs while other verbs are uncommon in one of the forms:

[1] The data and software used to extract the grammar we draw these examples from is described in Section 6.5.

Possessive rule	$NP \rightarrow$	the NN of the NNP	the NNP's NN
	$NP \rightarrow$	the NNS_1 made by NNS_2	the NNS_2's NNS_1
Dative shift	$VP \rightarrow$	give NN to NP	give NP the NN
	$VP \rightarrow$	provide NP_1 to NP_2	give NP_2 NP_1
Adv./adj. phrase move	$S/VP \rightarrow$	$ADVP$ they VBP	they VPB $ADVP$
	$S \rightarrow$	it is $ADJP$ VP	VP is $ADJP$
Verb particle shift	$VP \rightarrow$	VB NP up	VB up NP
Reduced relative clause	$SBAR/S \rightarrow$	although PRP VBP that	although PRP VBP
	$ADJP \rightarrow$	very JJ that S	JJ S
Partitive constructions	$NP \rightarrow$	CD of the NN	CD NN
	$NP \rightarrow$	all $DT\backslash NP$	all of the $DT\backslash NP$
Topicalization	$S \rightarrow$	NP, VP .	VP, NP .
Passivization	$SBAR \rightarrow$	that NP had VBN	which was VBN by NP
Light verbs	$VP \rightarrow$	take action $ADVP$	to act $ADVP$
	$VP \rightarrow$	TO take a decision PP	TO decide PP

Table 1: A selection of meaning-preserving transformations and hand-picked examples of syntactic paraphrases that our system extracts capturing these.

give *decontamination equipment* to *Japan*
give *Japan decontamination equipment*

provide *decontamination equipment* to *Japan*
? provide *Japan decontamination equipment*

Note how our system extracts a dative shift rule for *to give* and a rule that both shifts and substitutes a more appropriate verb for *to provide*.

The use of syntactic nonterminals in our paraphrase rules to capture complex transforms also makes it possible to impose constraints on their application. For comparison, as Madnani et al. (2007) do not impose any constraints on how the nonterminal X can be realized, their equivalent of the *topicalization* rule would massively overgeneralize:

$$S \rightarrow \quad X_1, X_2 \; . \quad | \quad X_2, X_1 \; .$$

Additional examples of transforms our use of syntax allows us to capture are the *adverbial phrase shift* and the *reduction of a relative clause*, as well as other phenomena listed in Table 1.

Unsurprisingly, syntactic information alone is not sufficient to capture all transformations. For instance it is hard to extract generic paraphrases for all instances of *passivization*, since our syntactic model currently has no means of representing the morphological changes that the verb undergoes:

the reactor *leaks* radiation
radiation *is leaking* from the reactor .

Still, for cases where the verb's morphology does not change, we manage to learn a rule:

the radiation that the reactor had *leaked*
the radiation which *leaked* from the reactor .

Another example of a deficiency in our synchronous grammar models are *light verb* constructs such as:

to take a *walk*
to *walk* .

Here, a noun is transformed into the corresponding verb – something our synchronous syntactic CFGs are not able to capture except through memorization.

Our survey shows that we are able to extract appropriately generic representations for a wide range of paraphrastic transformations. This is a surprising result which shows that bilingual parallel corpora can be used to learn sentential paraphrases, and that they are a viable alternative to other data sources like monolingual parallel corpora, which more obviously contain sentential paraphrases, but are scarce.

6 Text-to-Text Applications

The core of many text-to-text generation tasks is sentential paraphrasing, augmented with specific constraints or goals. Since our model borrows much of its machinery from statistical machine translation – a sentential rewriting problem itself – it is straightforward to use our paraphrase grammars to generate new sentences using SMT's decoding and parameter optimization techniques. Our framework can be adapted to many different text-to-text generation tasks. These could include text simplification, sen-

tence compression, poetry generation, query expansion, transforming declarative sentences into questions, and deriving hypotheses for textual entailment. Each individual text-to-text application requires that our framework be adapted in several ways, by specifying:

- A mechanism for extracting synchronous grammar rules (in this paper we argue that pivot-based paraphrasing is widely applicable).
- An appropriate set of rule-level features that capture information pertinent to the task (e.g. whether a rule simplifies a phrase).
- An appropriate "objective function" that scores the output of the model, i.e. a task-specific equivalent to the BLEU metric in SMT.
- A development set with examples of the sentential transformations that we are modeling.
- Optionally, a way of injecting task-specific rules that were not extracted automatically.

In the remainder of this section, we illustrate how our bilingually extracted paraphrases can be adapted to perform sentence compression, which is the task of reducing the length of sentence while preserving its core meaning. Most previous approaches to sentence compression focused only on the deletion of a subset of words from the sentence (Knight and Marcu, 2002). Our approach follows Cohn and Lapata (2008), who expand the task to include substitutions, insertions and reorderings that are automatically learned from parallel texts.

6.1 Feature Design

In Section 4 we discussed phrasal probabilities. While these help quantify how good a paraphrase is in general, they do not make any statement on task-specific things such as the change in language complexity or text length. To make this information available to the decoder, we enhance our paraphrases with four compression-targeted features. We add the count features c_{src} and c_{tgt}, indicating the number of words on either side of the rule as well as two difference features: $c_{dcount} = c_{tgt} - c_{src}$ and the analogously computed difference in the average word length in characters, c_{davg}.

6.2 Objective Function

Given our paraphrasing system's connection to SMT, the naive/obvious choice for parameter optimization would be to optimize for BLEU over a set of paraphrases, for instance parallel English reference translations for a machine translation task (Madnani et al., 2007). For a candidate C and a reference R, (with lengths c and r) BLEU is defined as:

$$\text{BLEU}_N(C,R) = \begin{cases} e^{(1-c/r)} \cdot e^{\sum_{n=1}^{N} \log w_n p_n} & \text{if } c/r \leq 1 \\ e^{\sum_{n=1}^{N} \log w_n p_n} & \text{otherwise} \end{cases},$$

where p_n is the modified n-gram precision of C against R, with typically $N = 4$ and $w_n = \frac{1}{N}$. The "brevity penalty" term $e^{(1-c/r)}$ is added to prevent short candidates from achieving perfect scores.

Naively optimizing for BLEU, however, will result in a trivial paraphrasing system heavily biased towards producing identity "paraphrases". This is obviously not what we are looking for. Moreover, BLEU does not provide a mechanism for directly specifying a per-sentence compression rate, which is desirable for the compression task.

Instead, we propose PRÉCIS, an objective function tailored to the text compression task:

$$\text{PRÉCIS}_{\lambda,\varphi}(I,C,R) = \begin{cases} e^{\lambda(\varphi - c/i)} \cdot \text{BLEU}(C,R) & \text{if } c/i \geq \varphi \\ \text{BLEU}(C,R) & \text{otherwise} \end{cases}$$

For an input sentence I, an output C and reference compression R (with lengths i, c and r), PRÉCIS combines the precision estimate of BLEU with an additional "verbosity penalty" that is applied to compressions that fail to meet a given target compression rate φ. We rely on the BLEU brevity penalty to prevent the system from producing overly aggressive compressions. The scaling term λ determines how severely we penalize deviations from φ. In our experiments we use $\lambda = 10$.

It is straightforward to find similar adaptations for other tasks. For text simplification, for instance, the penalty term can include a readability metric. For poetry generation we can analogously penalize outputs that break the meter (Greene et al., 2010).

6.3 Development Data

To tune the parameters of our paraphrase system for sentence compression, we need an appropriate cor-

pus of reference compressions. Since our model is designed to compress by paraphrasing rather than deletion, the commonly used deletion-based compression data sets like the Ziff-Davis corpus are not suitable. We have thus created a corpus of compression paraphrases. Beginning with 9570 tuples of parallel English–English sentences obtained from multiple reference translations for machine translation evaluation, we construct a parallel compression corpus by selecting the longest reference in each tuple as the source sentence and the shortest reference as the target sentence. We further retain only those sentence pairs where the compression rate cr falls in the range $0.5 < cr \leq 0.8$. From these, we randomly select 936 sentences for the development set, as well as 560 sentences for a test set that we use to gauge the performance of our system.

6.4 Grammar Augmentations

As we discussed in Section 5, the paraphrase grammar we induce is capable of representing a wide variety of transformations. However, the formalism and extraction method are not explicitly geared towards a compression application. For instance, the synchronous nature of our grammar does not allow us to perform deletions of constituents as done by Cohn and Lapata (2007)'s tree transducers. One way to extend the grammar's capabilities towards the requirements of a given task is by injecting additional rules designed to capture appropriate operations.

For the compression task, this could include adding rules to delete target-side nonterminals:

$$JJ \rightarrow JJ \mid \varepsilon$$

This would render the grammar asynchronous and require adjustments to the decoding process. Alternatively, we can generate rules that specifically delete particular adjectives from the corpus:

$$JJ \rightarrow \text{superfluous} \mid \varepsilon \ .$$

In our experiments we evaluate the latter approach by generating optional deletion rules for all adjectives, adverbs and determiners.

6.5 Experimental Setup

We extracted a paraphrase grammar from the French–English Europarl corpus (v5). The bitext was aligned using the Berkeley aligner and the English side was parsed with the Berkeley parser. We obtained the initial translation grammar using the SAMT toolkit (Venugopal and Zollmann, 2009).

Grammar	# Rules
total	42,353,318
w/o identity	23,641,016
w/o complex constituents	6,439,923
w/o complex const. & identity	5,097,250

Table 2: Number and distribution of rules in our paraphrase grammar. Note the significant number of identity paraphrases and rules with complex nonterminal labels.

The grammars we extract tend to be extremely large. To keep their size manageable, we only consider translation rules that have been seen more than 3 times and whose translation probability exceeds 10^{-4} for pivot recombination. Additionally, we only retain the top 25 most likely paraphrases of each phrase, ranked by a uniformly weighted combination of phrasal and lexical paraphrase probabilities.

We tuned the model parameters to our PRÉCIS objective function, implemented in the Z-MERT toolkit (Zaidan, 2009). For decoding we used the Joshua decoder (Li et al., 2010). The language model used in our paraphraser and the Clarke and Lapata (2008) baseline system is a Kneser-Ney discounted 5-gram model estimated on the Gigaword corpus using the SRILM toolkit (Stolcke, 2002).

6.6 Evaluation

To assess the output quality of the resulting sentence compression system, we compare it to two state-of-the-art sentence compression systems. Specifically, we compare against our implementation of Clarke and Lapata (2008)'s compression model which uses a series of constraints in an integer linear programming (ILP) solver, and Cohn and Lapata (2007)'s tree transducer toolkit (T3) which learns a synchronous tree substitution grammar (STSG) from paired monolingual sentences. Unlike SCFGs, the STSG formalism allows changes to the tree topology. Cohn and Lapata argue that this is a natural fit for sentence compression, since deletions introduce structural mismatches. We trained the T3 software[2] on the 936 $\langle \text{full}, \text{compressed} \rangle$ sentence pairs that comprise our development set. This is equivalent in size to the training corpora that Cohn and Lapata (2007) used (their training corpora ranged from

[2] www.dcs.shef.ac.uk/people/T.Cohn/t3/

882–1020 sentence pairs), and has the advantage of being in-domain with respect to our test set. Both these systems reported results outperforming previous systems such as McDonald (2006). To showcase the value of the adaptations discussed above, we also compare variants of our paraphrase-based compression systems: using Hiero instead of syntax, using syntax with or without compression features, using an augmented grammar with optional deletion rules.

We solicit human judgments of the compressions along two five-point scales: grammaticality and meaning. Judges are instructed to decide how much the meaning from a reference translation is retained in the compressed sentence, with a score of 5 indicating that all of the important information is present, and 1 being that the compression does not retain any of the original meaning. Similarly, a grammar score of 5 indicates perfect grammaticality, and a grammar score of 1 is assigned to sentences that are entirely ungrammatical. To ensure fairness, we perform pairwise system comparisons with compression rates strictly tied on the sentence-level. For any comparison, a sentence is only included in the computation of average scores if the difference between both systems' compression rates is < 0.05.[3]

Table 4 shows a set of pairwise comparisons for compression rates ≈ 0.5. We see that going from a Hiero-based to a syntactic paraphrase grammar yields a significant improvement in grammaticality. Adding compression-specific features improves grammaticality even further. Further augmenting the grammar with deletion rules significantly helps retain the core meaning at compression rates this high, however compared to the un-augmented syntactic system grammaticality scores drop. While our approach significantly outperforms the T3 system, we are not able to match ILP's results in grammaticality.

In Table 3 we compare our system to the ILP approach at a modest compression rate of ≈ 0.8. Here, we significantly outperform ILP in meaning retention while achieving comparable results in grammaticality. This improvement is significant at $p < 0.0001$, using the sign test, while the better grammaticality score of the ILP system is not statistically significant ($p < 0.088$). These results indicate that, over a variety of compression rates, our framework for text-to-text generation is performing as well as or better than specifically tailored state-of-the-art methods.

[3]Because evaluation quality correlates linearly with compression rate, the community-accepted practice of not comparing based on a closely tied compression rate is potentially subject to erroneous interpretation (Napoles et al., 2011).

	CR	Meaning	Grammar
Reference	0.73	4.26	4.35
Syntax+Feat.	0.80	**3.67**	3.38
ILP	0.80	3.50	3.49
Random Deletions	0.50	1.94	1.57

Table 3: Results of the human evaluation on longer compressions: pairwise compression rates (CR), meaning and grammaticality scores. Bold indicates a statistically significance difference at $p < 0.05$.

	CR	Meaning	Grammar
Hiero	0.56	2.57	2.35
Syntax	0.56	2.76	**2.67**
Syntax	0.53	2.70	2.49
Syntax+Feat.	0.53	2.71	**2.54**
Syntax+Feat.	0.54	2.79	**2.71**
Syntax+Aug.	0.54	**2.96**	2.52
Syntax+Aug.	0.52	2.87	2.40
ILP	0.52	2.83	**3.09**
Syntax+Aug.	0.50	2.41	**2.20**
T3	0.50	2.01	1.93

Table 4: Human evaluation for shorter compressions and for variations of our paraphrase system. +Feat. includes the compression features from Section 6.1, +Aug. includes optional deletion rules from Section 6.4.

Table 5 shows an example sentence drawn from our test set and the compressions produced by the different systems. We see that both the paraphrase and ILP systems produce good quality results, with the paraphrase system retaining the meaning of the source sentence more accurately.

7 Conclusion

In this work we introduced a method to learn syntactically informed paraphrases from bilingual parallel texts. We discussed the expressive power and limitations of our formalism and outlined straightforward adaptation strategies for applications in text-to-text generation. We demonstrated when our paraphrasing system was adapted to do sentence compression, it achieved results competitive with state-of-the-art compression systems with only minimal effort.

Source	he also expected that he would have a role in the future at the level of the islamic movement across the palestinian territories , even if he was not lucky enough to win in the elections .
Reference	he expects to have a future role in the islamic movement in the palestinian territories if he is not successful in the elections .
Syntax+Feat.	he also expected that he would have a role in the future of the islamic movement in the palestinian territories , although he was not lucky enough to win elections .
ILP	he also expected that he would have a role at the level of the islamic movement , even if he was not lucky enough to win in the elections .
Source	in this war which has carried on for the last 12 days , around 700 palestinians , which include a large number of women and children , have died .
Reference	about 700 palestinians , mostly women and children , have been killed in the israeli offensive over the last 12 days .
Syntax+Feat.	in this war has done for the last 12 days , around 700 palestinians , including women and children , died .
ILP	in this war which has carried for the days palestinians , which include a number of women and children died .
Source	hala speaks arabic most of the time with her son , taking into consideration that he can speak english with others .
Reference	hala speaks to her son mostly in arabic , as he can speak english to others .
Syntax+Feat.	hala speaks arabic most of the time with her son , considering that he can speak english with others .
ILP	hala speaks arabic most of the time , taking into consideration that he can speak english with others .

Table 5: Example compressions produced by the two systems in Table 3 for three input sentences from our test data.

Acknowledgments

We would like to thank Trevor Cohn for kindly providing us with the T3 compression system. This research was supported by the NSF under grant IIS-0713448. Opinions, interpretations, and conclusions are the authors' alone.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall.

Peter G. Anick and Suresh Tipirneni. 1999. The paraphrase search assistant: terminological feedback for iterative information seeking. In *Proceedings of SIGIR*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*.

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of EMNLP*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT/NAACL*.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of ACL*.

Regina Barzilay. 2003. *Information Fusion for Mutlidocument Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University, New York.

Rahul Bhagat and Deepak Ravichandran. 2008. Large scale acquisition of paraphrases for learning surface patterns. In *Proceedings of ACL/HLT*.

Peter Brown, John Cocke, Stephen Della Pietra, Vincent Della Pietra, Frederick Jelinek, Robert Mercer, and

Paul Poossin. 1990. A statistical approach to language translation. *Computational Linguistics*, 16(2), June.

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

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of EMNLP*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:273–381.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of EMNLP-CoLing*.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the COLING*.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research (JAIR)*, 34:637–674.

Peter W. Culicover. 1968. Paraphrase generation and information retrieval from stored text. *Mechanical Translation and Computational Linguistics*, 11(1-2):78–88.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the COLING*.

Mark Dras. 1997. Representing paraphrases using synchronous tree adjoining grammars. In *Proceedings of ACL*.

Mark Dras. 1999. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. Ph.D. thesis, Macquarie University, Australia.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL*.

William Gale and Kenneth Church. 1993. A program for aligning sentences in bilingual corpora. *Compuatational Linguistics*, 19(1):75–90.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL*.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of EMNLP*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of EMNLP*.

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*.

Raymond Kozlowski, Kathleen McCoy, and K. Vijay-Shanker. 2003. Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Workshop On Paraphrasing*.

Irene Langkilde and Kevin Knight. 1998. The practical value of n-grams in generation. In *Workshop On Natural Language Generation*, Ontario, Canada.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of WMT09*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Ann Irvine, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Ziyuan Wang, Jonathan Weese, and Omar Zaidan. 2010. Joshua 2.0: A toolkit for parsing-based machine translation with syntax, semirings, discriminative training and other goodies. In *Proceedings of WMT10*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules from text. *Natural Language Engineering*, 7(3):343–360.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment templates for statistical machine translation. In *Proceedings of the ACL/Coling*.

Nitin Madnani and Bonnie Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–388.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of WMT07*.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of EACL*.

Kathleen R. McKeown. 1979. Paraphrasing using given and new information in a question-answer system. In *Proceedings of ACL*.

Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of ACL*.

Marie W. Meteer and Varda Shaked. 1988. Strategies for effective paraphrasing. In *Proceedings of COLING*.

Kazunori Muraki. 1982. On a semantic model for multilingual paraphrasing. In *Proceedings of COLING*.

Courtney Napoles, Chris Callison-Burch, and Benjamin Van Durme. 2011. Evaluating sentence compression: Pitfalls and suggested remedies. In *Workshop on Monolingual Text-To-Text Generation*.

Franz Josef Och. 2003a. Minimum error rate training for statistical machine translation. In *Proceedings of ACL*.

Franz Josef Och. 2003b. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

Karolina Owczarzak, Declan Groves, Josef Van Genabith, and Andy Way. 2006. Contextual bitext-derived paraphrases in automatic MT evaluation. In *Proceedings of WMT06*.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT/NAACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of ACL*.

Deepak Ravichandran and Eduard Hovy. 2002. Learning sufrace text patterns for a question answering system. In *Proceedings of ACL*.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of ACL*.

Hadar Shemtov. 1996. Generation of paraphrases from ambiguous logical forms. In *Proceedings of COLING*.

Stuart Shieber and Yves Schabes. 1990. Generation and synchronous tree-adjoining grammars. In *Workshop On Natural Language Generation*.

Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2010. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In *Non-Transformational Syntax: Formal and Explicit Models of Grammar*. Wiley-Blackwell.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceeding of the International Conference on Spoken Language Processing*.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP*, Proceedings of EMNLP.

Ashish Venugopal and Andreas Zollmann. 2009. Grammar based statistical MT on Hadoop: An end-to-end toolkit for large scale PSCFG based MT. *Prague Bulletin of Mathematical Linguistics*, 91.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL*.

Kazuhide Yamamoto. 2002. Machine translation by interaction between paraphraser and transfer. In *Proceedings of COLING*.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008a. Combining multiple resources to improve SMT-based paraphrasing model. In *Proceedings of ACL/HLT*.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008b. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *Proceedings of ACL/HLT*.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of ACL*.

Liang Zhou, Chin-Yew Lin, Dragos Stefan Munteanu, and Eduard Hovy. 2006. Paraeval: Using paraphrases to evaluate summaries automatically. In *Proceedings of HLT/NAACL*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of WMT06*.

Joint Models for Chinese POS Tagging and Dependency Parsing

Zhenghua Li[†], Min Zhang[‡], Wanxiang Che[†], Ting Liu[†], Wenliang Chen[‡] and Haizhou Li[‡]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{lzh,car,tliu}@ir.hit.edu.cn
[‡]Institute for Infocomm Research, Singapore
{mzhang,wechen,hli}@i2r.a-star.edu.sg

Abstract

Part-of-speech (POS) is an indispensable feature in dependency parsing. Current research usually models POS tagging and dependency parsing independently. This may suffer from error propagation problem. Our experiments show that parsing accuracy drops by about 6% when using automatic POS tags instead of gold ones. To solve this issue, this paper proposes a solution by jointly optimizing POS tagging and dependency parsing in a unique model. We design several joint models and their corresponding decoding algorithms to incorporate different feature sets. We further present an effective pruning strategy to reduce the search space of candidate POS tags, leading to significant improvement of parsing speed. Experimental results on Chinese Penn Treebank 5 show that our joint models significantly improve the state-of-the-art parsing accuracy by about 1.5%. Detailed analysis shows that the joint method is able to choose such POS tags that are more helpful and discriminative from parsing viewpoint. This is the fundamental reason of parsing accuracy improvement.

1 Introduction

In dependency parsing, features consisting of part-of-speech (POS) tags are very effective, since pure lexical features lead to severe data sparseness problem. Typically, POS tagging and dependency parsing are modeled in a pipelined way. However, the pipelined method is prone to error propagation, especially for Chinese. Due to the lack of morphological features, Chinese POS tagging is even harder than other languages such as English. The state-of-the-art accuracy of Chinese POS tagging is about 93.5%, which is much lower than that of English (about 97% (Collins, 2002)). Our experimental results show that parsing accuracy decreases by about 6% on Chinese when using automatic POS tagging results instead of gold ones (see Table 3 in Section 5). Recent research on dependency parsing usually overlooks this issue by simply adopting gold POS tags for Chinese data (Duan et al., 2007; Zhang and Clark, 2008b; Huang and Sagae, 2010). In this paper, we address this issue by jointly optimizing POS tagging and dependency parsing.

Joint modeling has been a popular and effective approach to simultaneously solve related tasks. Recently, many successful joint models have been proposed, such as joint tokenization and POS tagging (Zhang and Clark, 2008a; Jiang et al., 2008; Kruengkrai et al., 2009), joint lemmatization and POS tagging (Toutanova and Cherry, 2009), joint tokenization and parsing (Cohen and Smith, 2007; Goldberg and Tsarfaty, 2008), joint named entity recognition and parsing (Finkel and Manning, 2009), joint parsing and semantic role labeling (SRL) (Li et al., 2010), joint word sense disambiguation and SRL (Che and Liu, 2010), joint tokenization and machine translation (MT) (Dyer, 2009; Xiao et al., 2010) and joint parsing and MT (Liu and Liu, 2010). Note that the aforementioned "parsing" all refer to *constituent parsing*.

As far as we know, there are few successful models for jointly solving *dependency parsing* and other tasks. Being facilitated by Conference on Computational Natural Language Learning (CoNLL) 2008 and 2009 shared tasks, several joint models of dependency parsing and SRL have been proposed. Nevertheless, the top-ranked systems all adopt pipelined approaches (Surdeanu et al., 2008;

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1180–1191,
Edinburgh, Scotland, UK, July 27–31, 2011.

Hajič et al., 2009). Theoretically, joint modeling of POS tagging and dependency parsing should be helpful to the two individual tasks. On the one hand, syntactic information can help resolve some POS ambiguities which are difficult to handle for the sequential POS tagging models. On the other hand, more accurate POS tags should further improve dependency parsing.

For joint POS tagging and dependency parsing, the major issue is to design effective decoding algorithms to capture rich features and efficiently search out the optimal results from a huge hypothesis space.[1] In this paper, we propose several dynamic programming (DP) based decoding algorithms for our joint models by extending existing parsing algorithms. We also present effective pruning techniques to speed up our decoding algorithms. Experimental results on Chinese Penn Treebank show that our joint models can significantly improve the state-of-the-art parsing accuracy by about 1.5%.

The remainder of this paper is organized as follows. Section 2 describes the pipelined method, including the POS tagging and parsing models. Section 3 discusses the joint models and the decoding algorithms, while Section 4 presents the pruning techniques. Section 5 reports the experimental results and error analysis. We review previous work closely related to our method in Section 6, and conclude this paper in Section 7.

2 The Baseline Pipelined Method

Given an input sentence $\mathbf{x} = w_1...w_n$, we denote its *POS tag sequence* by $\mathbf{t} = t_1...t_n$, where $t_i \in \mathcal{T}$, $1 \leq i \leq n$, and $\mathcal{T}$ is the POS tag set. A *dependency tree* is denoted by $\mathbf{d} = \{(h, m) : 0 \leq h \leq n, 0 < m \leq n\}$, where (h, m) represents a dependency $w_h \rightarrow w_m$ whose *head* word (or *father*) is w_h and *modifier* (or *child*) is w_m. w_0 is an artificial root token which is used to simplify the formalization of the problem.

The pipelined method treats POS tagging and dependency parsing as two cascaded problems. First, an optimal POS tag sequence $\hat{\mathbf{t}}$ is determined.

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} Score_{\text{pos}}(\mathbf{x}, \mathbf{t})$$

Then, an optimal dependency tree $\hat{\mathbf{d}}$ is determined based on $\mathbf{x}$ and $\hat{\mathbf{t}}$.

$$\hat{\mathbf{d}} = \arg\max_{\mathbf{d}} Score_{\text{syn}}(\mathbf{x}, \hat{\mathbf{t}}, \mathbf{d})$$

2.1 POS Tagging

POS tagging is a typical sequence labeling problem. Many models have been successfully applied to sequence labeling problems, such as maximum-entropy (Ratnaparkhi, 1996), conditional random fields (CRF) (Lafferty et al., 2001) and perceptron (Collins, 2002). We use perceptron to build our POS tagging baseline for two reasons. Firstly, as a linear model, perceptron is simple, fast, and effective. It is competitive to CRF in tagging accuracy but requires much less training time (Shen et al., 2007). Secondly, perceptron has been successfully applied to dependency parsing as well (Koo and Collins, 2010). In this paper, perceptron is used in all models including the POS tagging model, the dependency parsing models and the joint models.

In a perceptron, the score of a tag sequence is

$$Score_{\text{pos}}(\mathbf{x}, \mathbf{t}) = \mathbf{w}_{\text{pos}} \cdot \mathbf{f}_{\text{pos}}(\mathbf{x}, \mathbf{t})$$

where $\mathbf{f}_{\text{pos}}(\mathbf{x}, \mathbf{t})$ refers to the feature vector and $\mathbf{w}_{\text{pos}}$ is the corresponding weight vector.

For POS tagging features, we follow the work of Zhang and Clark (2008a). Three feature sets are considered: *POS unigram*, *bigram* and *trigram* features. For brevity, we will refer to the three sets as $\underline{w_i\ t_i}$, $\underline{t_{i-1}\ t_i}$ and $\underline{t_{i-2}\ t_{i-1}\ t_i}$.

Given $\mathbf{w}_{\text{pos}}$, we adopt the Viterbi algorithm to get the optimal tagging sequence.

2.2 Dependency Parsing

Recently, graph-based dependency parsing has gained more and more interest due to its state-of-the-art accuracy. Graph-based dependency parsing views the problem as finding the highest scoring tree from a directed graph. Based on dynamic programming decoding, it can efficiently find an optimal tree in a huge search space. In a graph-based model, the

[1] It should be noted that it is straightforward to simultaneously do POS tagging and constituent parsing, as POS tags can be regarded as non-terminals in the constituent structure (Levy and Manning, 2003). In addition, Rush et al. (2010) describes an efficient and simple inference algorithm based on dual decomposition and linear programming relaxation to combine a lexicalized constituent parser and a trigram POS tagger.

score of a dependency tree is factored into scores of small parts (subtrees).

$$Score_{\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) = \mathbf{w}_{\text{syn}} \cdot \mathbf{f}_{\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) = \sum_{p \subseteq \mathbf{d}} Score_{\text{syn}}(\mathbf{x}, \mathbf{t}, p)$$

where p is a scoring part which contains one or more dependencies in the dependency tree $\mathbf{d}$. Figure 1 shows different types of scoring parts used in current graph-based models.

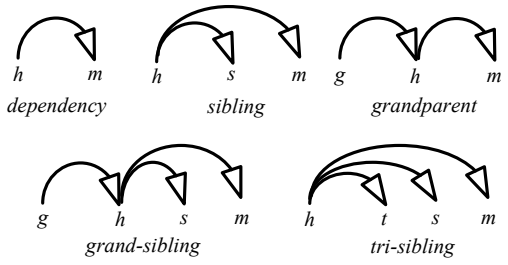


Figure 1: Different types of scoring parts used in current graph-based models (Koo and Collins, 2010).

Eisner (1996) proposes an $O(n^3)$ decoding algorithm for dependency parsing. Based on the algorithm, McDonald et al. (2005) propose the *first-order* model, in which the scoring parts only contains dependencies. The *second-order* model of McDonald and Pereira (2006) incorporates sibling parts and also needs $O(n^3)$ parsing time. The *second-order* model of Carreras (2007) incorporates both sibling and grandparent parts, and needs $O(n^4)$ parsing time. However, the grandparent parts are restricted to those composed of *outermost grandchildren*. Koo and Collins (2010) propose efficient decoding algorithms of $O(n^4)$ for *third-order* models. In their paper, they implement two versions of third-order models, Model 1 and Model 2 according to their naming. Model 1 incorporates only grand-sibling parts, while Model 2 incorporates both grand-sibling and tri-sibling parts. Their experiments on English and Czech show that Model 1 and Model 2 obtain nearly the same parsing accuracy. Therefore, we use Model 1 as our third-order model in this paper.

We use three versions of graph-based dependency parsing models.

- The first-order model (O1): the same with McDonald et al. (2005).
- The second-order model (O2): the same with Model 1 in Koo and Collins (2010), but without using grand-sibling features.[2]
- The third-order model (O3): the same with Model 1 in Koo and Collins (2010).

We adopt linear models to define the score of a dependency tree. For the third-order model, the score of a dependency tree is represented as:

$$\begin{aligned} Score_{\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) = & \sum_{\{(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{\text{dep}} \cdot \mathbf{f}_{\text{dep}}(\mathbf{x}, \mathbf{t}, h, m) \\ & + \sum_{\{(h,s)(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{\text{sib}} \cdot \mathbf{f}_{\text{sib}}(\mathbf{x}, \mathbf{t}, h, s, m) \\ & + \sum_{\{(g,h),(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{\text{grd}} \cdot \mathbf{f}_{\text{grd}}(\mathbf{x}, \mathbf{t}, g, h, m) \\ & + \sum_{\{(g,h),(h,s),(h,m)\} \subseteq \mathbf{d}} \mathbf{w}_{\text{gsib}} \cdot \mathbf{f}_{\text{gsib}}(\mathbf{x}, \mathbf{t}, g, h, s, m) \end{aligned}$$

For the first- and second-order models, the above formula is modified by deactivating extra parts.

For parsing features, we follow standard practice for graph-based dependency parsing (McDonald, 2006; Carreras, 2007; Koo and Collins, 2010). Since these features are highly related with our joint decoding algorithms, we summarize the features as follows.

- Dependency Features, $\mathbf{f}_{\text{dep}}(\mathbf{x}, \mathbf{t}, h, m)$
  - Unigram Features: $\underline{w_h t_h\ dir}$, $\underline{w_m t_m\ dir}$
  - Bigram Features: $\underline{w_h t_h\ w_m t_m\ dir\ dist}$
  - In Between Features: $\underline{t_h\ t_b\ t_m\ dir\ dist}$
  - Surrounding Features: $\underline{t_{h-1}\ t_h\ t_{h+1}\ t_{m-1}\ t_m\ t_{m+1}\ dir\ dist}$
- Sibling Features, $\mathbf{f}_{\text{sib}}(\mathbf{x}, \mathbf{t}, h, s, m)$ $\underline{w_h\ t_h\ w_s\ t_s\ w_m\ t_m\ dir}$
- Grandparent Features, $\mathbf{f}_{\text{grd}}(\mathbf{x}, \mathbf{t}, g, h, m)$ $\underline{w_g\ t_g\ w_h\ t_h\ w_m\ t_m\ dir\ gdir}$
- Grand-sibling Features, $\mathbf{f}_{\text{gsib}}(\mathbf{x}, \mathbf{t}, g, h, s, m)$ $\underline{w_g\ t_g\ w_h\ t_h\ w_s\ t_s\ w_m\ t_m\ dir\ gdir}$

[2]This second-order model incorporates grandparent features composed of all grandchildren rather than just outermost ones, and outperforms the one of Carreras (2007) according to the results in Koo and Collins (2010).

where b denotes an index between h and m; dir and $dist$ are the direction and distance of (h, m); $gdir$ is the direction of (g, h). We also use back-off features by generalizing from very specific features over word forms, POS tags, directions and distances to less sparse features over just POS tags or considering fewer nodes. To avoid producing too many sparse features, at most two word forms are used at the same time in sibling, grandparent and grand-sibling features, while POS tags are used instead for other nodes; meanwhile, at most four POS tags are considered at the same time for surrounding features.

3 Joint Models

In the joint method, we aim to simultaneously solve the two problems.

$$(\hat{\mathbf{t}}, \hat{\mathbf{d}}) = \arg\max_{\mathbf{t},\mathbf{d}} Score_{\text{joint}}(\mathbf{x}, \mathbf{t}, \mathbf{d})$$

Under the linear model, the score of a tagged dependency tree is:

$$\begin{aligned} Score_{\text{joint}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) &= Score_{\text{pos}}(\mathbf{x}, \mathbf{t}) \\ &\quad + Score_{\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) \\ &= \mathbf{w}_{\text{pos}\oplus\text{syn}} \cdot \mathbf{f}_{\text{pos}\oplus\text{syn}}(\mathbf{x}, \mathbf{t}, \mathbf{d}) \end{aligned}$$

where $\mathbf{f}_{\text{pos}\oplus\text{syn}}(.)$ means the concatenation of $\mathbf{f}_{\text{pos}}(.)$ and $\mathbf{f}_{\text{syn}}(.)$. Under the joint model, the weights of POS and syntactic features, $\mathbf{w}_{\text{pos}\oplus\text{syn}}$, are simultaneously learned. We expect that POS and syntactic features can interact each other to determine an optimal joint result.

Similarly to the baseline dependency parsing models, we define the *first-*, *second-*, and *third-order* joint models according to the syntactic features contained in $\mathbf{f}_{\text{syn}}(.)$.

In the following, we propose two versions of joint models which can capture different feature sets and have different complexity.

3.1 Joint Models of Version 1

The crucial problem for the joint method is to design effective decoding algorithms to capture rich features and efficiently search out the optimal results from a huge hypothesis space. Eisner (2000) describes a preliminary idea to handle *polysemy* by extending parsing algorithms. Based on this idea, we extend decoding algorithms of McDonald et al. (2005) and Koo and Collins (2010), and propose two DP based decoding algorithms for our joint models of version 1.

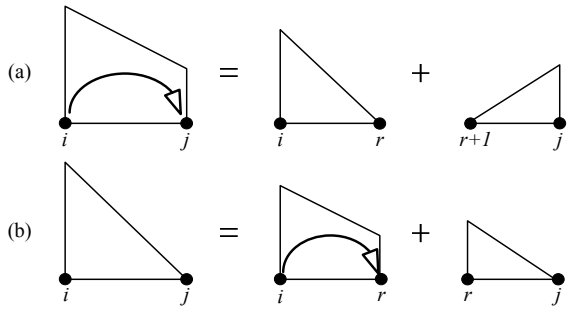


Figure 2: The DP structures and derivations of the first-order decoding algorithm of joint models of version 1. We omit symmetric right-headed versions for brevity. Trapezoids denote *incomplete spans*. Triangles denote *complete spans*. Solid circles denote POS tags of the corresponding indices.

The decoding algorithm of O1: As shown in Figure 2, the first-order joint decoding algorithm utilizes two types of dynamic programming structures. (1) ***I**ncomplete spans* consist of a dependency and the region between the head and modifier; (2) ***C**omplete spans* consist of a headword and its descendants on one side. Each span is recursively created by combining two smaller and adjacent spans in a bottom-up fashion.

The pseudo codes are given in Algorithm 1. $I_{(i,j)(t_i,t_j)}$ denotes an incomplete span from i to j whose boundary POS tags are t_i and t_j. $C_{(i,j)(t_i,t_j)}$ refers to a complete span from i to j whose boundary POS tags are t_i and t_j. Conversely, $I_{(j,i)(t_j,t_i)}$ and $C_{(j,i)(t_j,t_i)}$ represent spans of the other direction. Note that in these notations the first argument index always refers to the *head* of the span.

Line 6 corresponds to the derivation in Figure 2-(a). $Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_{r+1}, t_j, p = \{(i, j)\})$ captures the joint features invented by this combination, where $p = \{(i, j)\}$ means that the newly observed scoring part is the dependency (i, j). The syntactic features, denoted by $\mathbf{f}_{\text{syn}}(\mathbf{x}, t_i, t_j, i, j)$, can only incorporate *syntactic unigram and bigram* features. The surrounding and in between features are unavailable, because the context POS tags, such as t_b and t_{i-1}, are not contained in the DP struc-

Algorithm 1 The first-order joint decoding algorithm of version 1

1: $\forall 0 \le i \le n, t_i \in \mathcal{T}\ \ C_{(i,i)(t_i,t_i)} = 0$ ◁ initialization
2: **for** $w = 1..n$ **do** ◁ span width
3: **for** $i = 0..(n-w)$ **do** ◁ span start index
4: $j = i + w$ ◁ span end index
5: **for** $(t_i, t_j) \in \mathcal{T}^2$ **do**
6: $I_{(i,j)(t_i,t_j)} = \max_{i \le r < j} \max_{(t_r,t_{r+1}) \in \mathcal{T}^2} \{C_{(i,r)(t_i,t_r)} + C_{(j,r+1)(t_j,t_{r+1})} + Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_{r+1}, t_j, p = \{(i,j)\})\}$
7: $I_{(j,i)(t_j,t_i)} = \max_{i \le r < j} \max_{(t_r,t_{r+1}) \in \mathcal{T}^2} \{C_{(i,r)(t_i,t_r)} + C_{(j,r+1)(t_j,t_{r+1})} + Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_{r+1}, t_j, p = \{(j,i)\})\}$
8: $C_{(i,j)(t_i,t_j)} = \max_{i < r \le j} \max_{t_r \in \mathcal{T}} \{I_{(i,r)(t_i,t_r)} + C_{(r,j)(t_r,t_j)} + Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_j, p = \emptyset)\}$
9: $C_{(j,i)(t_j,t_i)} = \max_{i \le r < j} \max_{t_r \in \mathcal{T}} \{C_{(r,i)(t_r,t_i)} + I_{(j,r)(t_j,t_r)} + Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_j, p = \emptyset)\}$
10: **end for**
11: **end for**
12: **end for**

tures. Therefore, we adopt ***pseudo surrounding and in between features*** by simply fixing the context POS tags as the single most likely ones (McDonald, 2006). Taking the in between features as an example, we use $\underline{t_i\ \hat{t}_b\ t_j\ dir\ dist}$ instead, where $\hat{t}_b$ is the 1-best tag determined by the baseline POS tagger. The POS features, denoted by $\mathbf{f}_{\text{pos}}(\mathbf{x}, t_i, t_r, t_{r+1}, t_j)$, can only incorporate all *POS unigram and bigram* features.[3] Similarly, we use ***pseudo POS trigram features*** such as $\underline{\hat{t}_{r-1}\ t_r\ t_{r+1}}$.

Line 8 corresponds to the derivation in Figure 2-(b). Since this combination invents no scoring part ($p = \emptyset$), $Score_{\text{joint}}(\mathbf{x}, t_i, t_r, t_j, p = \emptyset)$ is only composed of POS features.[4]

Line 7 and Line 9 create spans in the opposite direction, which can be analogously illustrated. The space and time complexity of the algorithm are respectively $O(n^2q^2)$ and $O(n^3q^4)$, where $q = |\mathcal{T}|$.[5]

The decoding algorithm of O2 & O3: Figure 3 illustrates the second- and third-order decoding algorithm of joint models of version 1. A new kind of span, named the *sibling span*, is used to capture sibling structures. Furthermore, each span is augmented with a grandparent-index to capture both grandparent and grand-sibling structures. It is straightforward to derive the pseudo codes of the al-

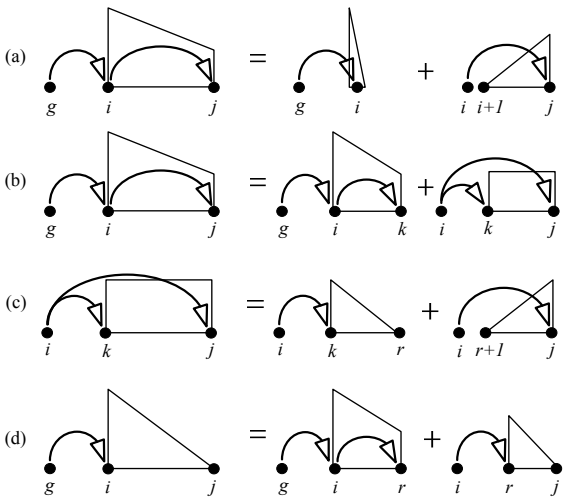


Figure 3: The DP structures and derivations of the second- and third-order joint decoding algorithm of version 1. For brevity, we elide the right-headed and right-grandparented versions. Rectangles represent sibling spans.

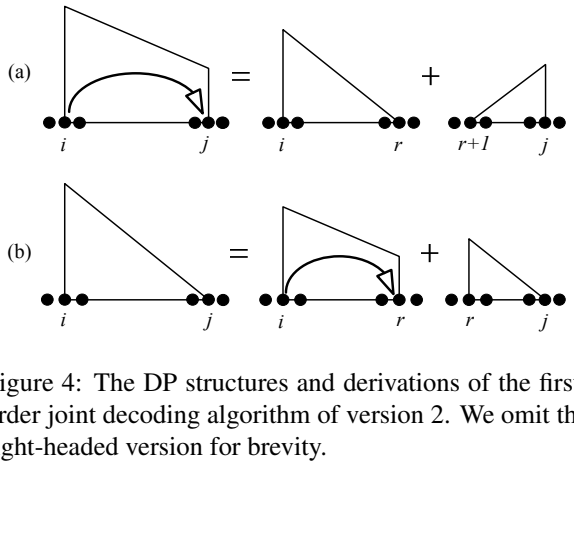


Figure 4: The DP structures and derivations of the first-order joint decoding algorithm of version 2. We omit the right-headed version for brevity.

[3] ① $\underline{w_r\ t_r}$ if $i \ne r$; ② $\underline{w_{r+1}\ t_{r+1}}$ if $r+1 \ne j$; ③ $\underline{t_r\ t_{r+1}}$ if $r \ne i$ or $r+1 \ne j$; ④ $\underline{t_i\ t_r}$ if $r-1 = i$; ⑤ $\underline{t_{r+1}\ t_j}$ if $r+2 = j$. Note that $\underline{w_i\ t_i}$, $\underline{w_j\ t_j}$ and $\underline{t_i\ t_j}$ (if $i = j-1$) are not incorporated here to avoid double counting.

[4] ① $\underline{w_r\ t_r}$ if $r \ne j$; ② $\underline{t_i\ t_r}$ if $i = r-1$; ③ $\underline{t_r\ t_j}$ if $r+1 = j$. Pseudo trigram features can be added accordingly.

[5] We can reduce the time complexity to $O(n^3q^3)$ by strictly adopting the DP structures in the parsing algorithm of Eisner (1996). However, that may make the algorithm harder to comprehend.

gorithm from Figure 3. We omit them due to space limitation. Pseudo surrounding, in between and POS trigram features are used due to the same reason as above. The space and time complexity of the algorithm are respectively $O(n^3q^3)$ and $O(n^4q^5)$.

3.2 Joint Models of Version 2

To further incorporate genuine syntactic surrounding and POS trigram features in the DP structures, we extend the algorithms of joint models of version 1, and propose our joint models of version 2.

The decoding algorithm of O1: Figure 4 illustrates the first-order joint decoding algorithm of version 2. Compared with the structures in Figure 2, each span is augmented with the POS tags surrounding the boundary indices. These context POS tags enable $Score_{\text{joint}}(.)$ in line 6-9 of Algorithm 1 to capture the ***syntactic surrounding and POS trigram features***, but also require enumeration of POS tags over more indices. For brevity, we skip the pseudo codes which can be easily derived from Algorithm 1. The space and time complexity of the algorithm are respectively $O(n^2q^6)$ and $O(n^3q^{10})$.

The decoding algorithm of O2 & O3: Using the same idea as above, the second- and third-order joint decoding algorithms of version 2 can be derived based on Figure 3. Again, we omit both its DP structures and pseudo codes for the sake of brevity. Its space and time complexity are respectively $O(n^3q^7)$ and $O(n^4q^{11})$.

In between features, which should be regarded as non-local features in the joint situation, still cannot be incorporated in our joint models of version 2. Again, we adopt the pseudo version.

3.3 Comparison

Based on the above illustration, we can see that joint models of version 1 are more efficient with regard to the number of POS tags for each word, but fail to incorporate syntactic surrounding features and POS trigram features in the DP structures. On the contrary, joint models of version 2 can incorporate both aforementioned feature sets, but have higher complexity. These two versions of models will be thoroughly compared in the experiments.

4 Pruning Techniques

In this section, we introduce two pruning strategies to constrain the search space of our models due to their high complexity.

4.1 POS Tag Pruning

The time complexity of the joint decoding algorithm is unbearably high with regard to the number of candidate POS tags for each word ($q = |\mathcal{T}|$). We find that it would be extremely time-consuming even when we only use two most likely POS tags for each word ($q = 2$) even for joint models of version 1. To deal with this problem, we propose a pruning method that can effectively reduce the POS tag space based on a probabilistic tagging model.

We adopt a conditional log-linear model (Lafferty et al., 2001), which defines a conditional distribution of a POS tag sequence $\mathbf{t}$ given $\mathbf{x}$:

$$P(\mathbf{t}|\mathbf{x}) = \frac{e^{\mathbf{w}_{\text{pos}} \cdot \mathbf{f}_{\text{pos}}(\mathbf{x},\mathbf{t})}}{\sum_{\mathbf{t}} e^{\mathbf{w}_{\text{pos}} \cdot \mathbf{f}_{\text{pos}}(\mathbf{x},\mathbf{t})}}$$

We use the same feature set $\mathbf{f}_{\text{pos}}$ defined in Section 2.1, and adopt the *exponentiated gradient* algorithm to learn the weight vector $\mathbf{w}_{\text{pos}}$ (Collins et al., 2008).

The marginal probability of tagging a word w_i as t is

$$P(t_i = t|\mathbf{x}) = \sum_{\mathbf{t}:\mathbf{t}[i]\equiv t} P(\mathbf{t}|\mathbf{x})$$

which can be efficiently computed using the *forward-backward* algorithm.

We define $\texttt{pmax}_i(\mathbf{x})$ to be the highest marginal probability of tagging the word w_i:

$$\texttt{pmax}_i(\mathbf{x}) = \max_{t \in \mathcal{T}} P(t_i = t|\mathbf{x})$$

We then define the allowable candidate POS tags of the word w_i to be

$$\mathcal{T}_i(\mathbf{x}) = \{t : t \in \mathcal{T}, P(t_i = t|\mathbf{x}) \geq \lambda_t \times \texttt{pmax}_i(\mathbf{x})\}$$

where λ_t is the pruning threshold. $\mathcal{T}_i(\mathbf{x})$ is used to constrain the POS search space by replacing $\mathcal{T}$ in Algorithm 1.

4.2 Dependency Pruning

The parsing time grows quickly for the second- and third-order models (both baseline and joint) when the input sentence gets longer ($O(n^4)$). Following Koo and Collins (2010), we eliminate unlikely dependencies using a form of coarse-to-fine pruning (Charniak and Johnson, 2005; Petrov and Klein, 2007). On the development set, 68.87% of the dependencies are pruned, while the oracle dependency accuracy is 99.77%. We use 10-fold cross validation to do pruning on the training set.

5 Experiments

We use the Penn Chinese Treebank 5.1 (CTB5) (Xue et al., 2005). Following the setup of Duan et al. (2007), Zhang and Clark (2008b) and Huang and Sagae (2010), we split CTB5 into training (secs 001-815 and 1001-1136), development (secs 886-931 and 1148-1151), and test (secs 816-885 and 1137-1147) sets. We use the head-finding rules of Zhang and Clark (2008b) to turn the bracketed sentences into dependency structures.

We use the standard *tagging **acc**uracy* to evaluate POS tagging. For dependency parsing, we use ***word** accuracy* (also known as *dependency accuracy*), ***root** accuracy* and ***compl**ete match rate* (all excluding punctuation) .

For the averaged training, we train each model for 15 iterations and select the parameters that perform best on the development set.

5.1 Results of POS Tag Pruning

Figure 5 shows the distribution of words with different number of candidate POS tags and the k-best oracle tagging accuracy under different λ_t. To avoid dealing with words that have many candidate POS tags, we further apply a hard criterion that the decoding algorithms only consider top k candidate POS tags.

To find the best λ_t, we train and evaluate the second-order joint model of version 1 on the training and development sets pruned with different λ_t (top k = 5). We adopt the second-order joint model of version 1 because of its efficiency compared with the third-order models and its capability of capturing rich features compared with the first-order models. The results are shown in Table 1. The model

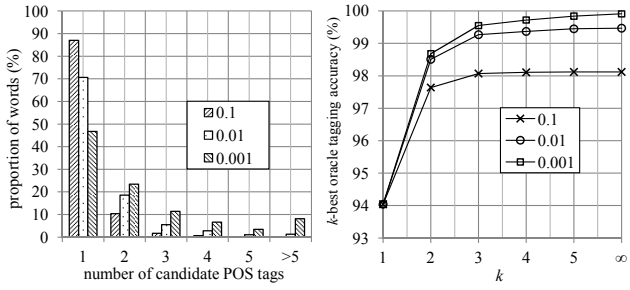


Figure 5: Results of POS tag pruning with different pruning threshold λ_t on the development set.

λ_t	word	root	compl.	acc.	speed
0.1	81.53	76.88	30.00	**94.17**	2.5
0.01	**81.83**	76.62	**30.62**	93.16	1.2
0.001	81.73	**77.38**	30.50	93.41	0.5

Table 1: Performance of the second-order joint model of version 1 with different pruning threshold λ_t (top k = 5) on the development set. "Acc." means the tagging accuracy. "Speed" refers to the parsing speed (the number of sentences processed per second).

with $\lambda_t = 0.1$ obtains the highest tagging accuracy, which is much higher than that of both $\lambda_t = 0.01$ and $\lambda_t = 0.001$. However, its parsing accuracy is inferior to the other two. $\lambda_t = 0.01$ produces slightly better parsing accuracy than $\lambda_t = 0.001$, and is twice faster. Finally, we choose $\lambda_t = 0.01$ due to the efficiency factor and our priority over the parsing accuracy.

Then we do experiments to find an optimal top k. Table 2 shows the results. We decide to choose $k = 3$ since it leads to best parsing accuracy.

From Table 1 and 2, we can have an interesting finding: it seems that the harder we filter the POS tag space, the higher tagging accuracy we get. In other words, **giving the joint model less flexibility of choosing POS tags leads to better tagging performance.**

Due to time limitation, we do not tune λ_t and k for other joint models. Instead, we simply adopt $\lambda_t = 0.01$ and top $k = 3$.

5.2 Final Results

Table 3 shows the final results on the test set. We list a few state-of-the-art results in the bottom. Duan07 refers to the results of Duan et al. (2007). They enhance the transition-based parsing model with

		Syntactic Metrics			Tagging Accuracy			Parsing Speed
		word	root	compl.	all-word	known	unknown	$Sent/Sec$
Joint Models V2	O3	**80.79**	**75.84**	**29.11**	92.80	93.88	76.80	0.3
	O2	80.49	75.49	28.24	92.68	93.77	76.27	0.6
	O1	77.37	68.64	23.09	**92.96**	94.05	76.64	2.0
Joint Models V1	O3	80.69	**75.90**	**29.06**	92.89	93.96	76.80	0.5
	O2	**80.74**	75.80	28.24	93.08	94.11	77.53	1.7
	O1	77.38	69.69	22.62	**93.20**	94.23	77.76	8.5
Auto POS	O3	**79.29**	74.65	**27.24**	93.51	94.36	80.78	2.0
	O2	79.03	**74.70**	27.19				5.8
	O1	75.68	68.06	21.10				17.4
	MSTParser2	77.95	72.04	25.50				4.1
	MSTParser1	75.84	68.55	21.36				5.2
	MaltParser	75.24	65.92	23.19				2.6
Gold POS	O3	**86.00**	77.59	**34.02**	100.0	100.0	100.0	-
	O2	**86.18**	**78.58**	**34.07**				-
	O1	82.24	70.10	26.02				-
	MSTParser2	85.24	77.41	33.19				-
	MSTParser1	83.04	71.49	27.59				-
	MaltParser	82.62	69.34	29.06				-
	H&S10	85.20	**78.32**	**33.72**				-
	Z&C08 single	84.33	76.73	32.79				-
	Z&C08 hybrid	**85.77**	76.26	**34.41**				-
	Duan07	83.88	73.70	32.70				-

Table 3: Final results on the test set. "Gold POS" means that gold POS tags are used as input by the pipelined parsing models; while "Auto POS" means that the POS tags are generated by the baseline POS tagging model.

top k	word	root	compl.	acc.	speed
2	81.46	76.12	30.50	**93.51**	2.7
3	**82.11**	**76.75**	29.75	93.31	1.7
4	81.75	76.62	30.38	93.25	1.4
5	81.83	76.62	**30.62**	93.16	1.2

Table 2: Performance of the second-order joint model of version 1 with different top k ($\lambda_t = 0.01$) on the development set.

the beam search. H&S10 refers to the results of Huang and Sagae (2010). They greatly expand the search space of the transition-based model by merging equivalent states with dynamic programming. Z&C08 refers to the results of Zhang and Clark (2008b). They use a hybrid model to combine the advantages of both graph-based and transition-based models. We also do experiments with two publicly available and widely-used parsers, MSTParser[6] and MaltParser[7]. MSTParser1 refers to the first-order graph-based model of McDonald et al. (2005), while MSTParser2 is the second-order model of McDonald and Pereira (2006). MaltParser is a transition-based parsing system. It integrates a number of classification algorithms and transition strategies. We adopt the support vector machine classifier and the arc-standard strategy (Nivre, 2008).

We can see that when using gold tags, our pipelined second- and third-order parsing models achieve best parsing accuracy, which is even higher than the hybrid model of Zhang and Clark (2008b). It is a little surprising that the second-order model slightly outperforms the third-order one. This may be possible, since Koo and Collins (2010) shows that the third-order model outperforms the second-order one by only 0.32% on English and 0.07% on Czech. In addition, we only use basic third-order features.

Both joint models of version 1 and 2 can consistently and significantly improve the parsing accuracy by about 1.5% for all first-, second- and third-order cases. Accidentally, the parsing accuracy of the second-order joint model of version 2 is lower

[6]http://sourceforge.net/projects/mstparser/

[7]http://maltparser.org/

error pattern	#	↓	error pattern	#	↑
DEC → DEG	237	114	NR → NN	184	100
NN → VV	389	73	NN → NR	106	91
DEG → DEC	170	39	NN → JJ	95	70
VV → NN	453	27	VA → VV	29	41
P → VV	52	24	JJ → NN	126	29
P → CC	39	13	VV → VA	67	10

Table 4: Error analysis of POS tagging. # means the error number of the corresponding pattern made by the baseline tagging model. ↓ and ↑ mean the error number reduced or increased by the joint model.

than that of its counterparts by about 0.3%. More experiments and further analysis may be needed to find out the reason. The two versions of joint models performs nearly the same, which indicates that using pseudo surrounding and POS trigram features may be sufficient for the joint method on this data set. In summary, we can conclude that **the joint framework is certainly helpful for dependency parsing.**

It is clearly shown in Table 3 that **the joint method surprisingly hurts the tagging accuracy**, which diverges from our discussion in Section 1. Some insights into this issue will be given in Section 5.3. Moreover, it seems that **the more syntactic features the joint method incorporates (from O1 to O3), the more the tagging accuracy drops**. We suspect that this is because the joint models are dominated by the syntactic features. Take the first-order joint model as an example. The dimension of the syntactic features $\mathbf{f}_{\text{syn}}$ is about 3.5 million, while that of $\mathbf{f}_{\text{pos}}$ is only about 0.5 million. The gap becomes much larger for the second- and third-order cases.

Comparing the parsing speed, we can find that the pruning of POS tags is very effective. The second-order joint model of version 1 can parse 1.7 sentences per second, while the pipelined second-order parsing model can parse 5.8 sentences per second, which is rather close considering that there is a factor of q^5.

5.3 Error Analysis

To find out the impact of our joint models on the individual tasks, we conduct detailed error analysis through comparing the results of the pipelined second-order parsing model and the second-order joint model of version 1.

Impact on POS tagging: Table 4 shows how the joint model changes the quantity of POS tagging error patterns compared with the pipelined model. An error pattern "X → Y" means that the focus word, whose true tag is 'X', is assigned a tag 'Y'. We choose these patterns with largest reduction or increase in the error number, and rank them in descending order of the variation.

From the left part of Table 4, we can see that the joint method is clearly better at resolving tagging ambiguities like {VV, NN} and {DEG, DEC}.[8] One common characteristic of these ambiguous pairs is that the local or even whole syntactic structure will be destructed if the wrong tag is chosen. In other words, resolving these ambiguities is critical and helpful from the parsing viewpoint. From another perspective, the joint model is capable of preferring the right tag with the help of syntactic structures, which is impossible for the baseline sequential labeling model.

In contrast, pairs like {NN, NR}, {VV, VA} and {NN, JJ} only slightly influence the syntactic structure when mis-tagged. The joint method performs worse on these ambiguous pairs, as shown in the right part of Table 4.

Impact on parsing: Table 5 studies the change of parsing error rates between the pipelined and joint model on different POS tag patterns. We present the most typical and prominent patterns in the table, and rank them in descending order of X's frequency of occurrence. We also show the change of proportion of different patterns, which is consistent with the results in Table 4.

From the table, we can see the joint model can achieve a large error reduction (0.8∼4.0%) for all the patterns "X → X". In other words, **the joint model can do better given the correct tags** than the pipelined method.

For all the patterns marked by ♢, except for the ambiguous pair {NN, JJ} (which we find is difficult to explain even after careful result analysis), the joint model also reduces the error rates (2.2∼15.4%). As

[8] DEG and DEC are the two POS tags for the frequently used auxiliary word "的" (dē, of) in Chinese. The associative "的" is tagged as DEG, such as "父亲/father 的 眼睛/eyes (eyes of the father)"; while the one in a relative clause is tagged as DEC, such as "他/he 取得/made 的 进步/progress (progress that he made)".

pattern	pipelined		joint	
	prop (%)	error (%)	prop (%)	error (%)
NN → NN	94.6	16.8	-1.1	-1.8
→ VV ♡	2.9	55.5	-0.5	+15.1
→ NR ♢	0.8	24.5	+0.7	-2.2
→ JJ ♢	0.7	17.9	+0.5	*+2.1*
VV → VV	89.6	34.2	-0.3	–4.0
→ NN ♡	6.6	66.4	-0.4	+0.7
→ VA ♢	1.0	38.8	+0.1	-15.4
NR → NR	91.7	15.4	-3.7	-0.8
→ NN ♢	5.9	21.7	+3.2	-3.7
P → P	92.8	22.6	+3.4	-3.2
→ VV ♡	3.0	50.0	-1.4	+10.7
→ CC ♡	2.3	74.4	-0.7	+21.9
JJ → JJ	80.5	11.2	-2.8	-2.0
→ NN ♢	9.8	18.3	+2.2	*+1.8*
DEG → DEG	86.5	11.1	+2.8	-3.6
→ DEC ♡	13.5	61.8	-3.1	+37.4
DEC → DEC	79.7	17.2	+12.1	-4.0
→ DEG ♡	20.2	56.5	-9.7	+40.2

Table 5: Comparison of parsing error rates on different POS tag patterns between the pipelined and joint models. Given a pattern "X → Y", "prop" means its *proportion* in all occurrence of 'X' ($\frac{Count(X \rightarrow Y)}{Count(X)}$), and "error" refers to its *parsing error rate* ($\frac{Count(\texttt{wrongly headed } X \rightarrow Y)}{Count(X \rightarrow Y)}$). The last two columns give the absolute reduction (-) or increase (+) in proportion and error rate made by the joint model. ♡ marks the patterns appearing in the left part of Table 4, while ♢ marks those in the right part of Table 4.

discussed earlier, these patterns concern ambiguous tag pairs which usually play similar roles in syntactic structures. This demonstrates that **the joint model can do better on certain tagging error patterns**.

For patterns marked by ♡, the error rate of the joint model usually increases by large margin. However, the proportion of these patterns is substantially decreased, since the joint model can better resolve these ambiguities with the help of syntactic knowledge.

In summary, we can conclude that the joint model is able to choose such POS tags that are more helpful and discriminative from parsing viewpoint. This is the fundamental reason of the parsing performance improvement.

6 Related Work

Theoretically, Eisner (2000) proposes a preliminary idea of extending the decoding algorithm for dependency parsing to handle polysemy. Here, word senses can be understood as POS-tagged words. Koo and Collins (2010) also briefly discuss that their third-order decoding algorithm can be modified to handle word senses using the idea of Eisner (2000).

In his PhD thesis, McDonald (2006) extends his second-order model with the idea of Eisner (2000) to study the impact of POS tagging errors on parsing accuracy. To make inference tractable, he uses top 2 candidate POS tags for each word based on a maximum entropy tagger, and adopts the single most likely POS tags for the surrounding and in between features. He conducts primitive experiments on English Penn Treebank, and shows that parsing accuracy can be improved from 91.5% to 91.9%. However, he finds that the model is unbearably time-consuming.

7 Conclusions

In this paper, we have systematically investigated the issue of joint POS tagging and dependency parsing. We propose and compare several joint models and their corresponding decoding algorithms which can incorporate different feature sets. We also propose an effective POS tag pruning method which can greatly improve the decoding efficiency. The experimental results show that our joint models can significantly improve the state-of-the-art parsing accuracy by more than 1.5%. Detailed error analysis shows that the fundamental reason for the parsing accuracy improvement is that the joint method is able to choose POS tags that are helpful and discriminative from parsing viewpoint.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by National Natural Science Foundation of China (NSFC) via grant 60803093, 60975055, the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology (HIT.NSRIF.2009069) and the Fundamental Research Funds for the Central Universities (HIT.KLOF.2010064).

References

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of*

EMNLP/CoNLL, pages 141–150.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL-05*, pages 173–180.

Wanxiang Che and Ting Liu. 2010. Jointly modeling wsd and srl with markov logic. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 161–169.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP-CoNLL 2007*, pages 208–217.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *JMLR*, 9:1775–1822.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP 2002*.

Xiangyu Duan, Jun Zhao, , and Bo Xu. 2007. Probabilistic models for action-based Chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 406–414.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING 1996*, pages 340–345.

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

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, pages 371–379, Columbus, Ohio, June. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL 2009*.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala, Sweden, July. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint chinese word segmentation and pos tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 513–521.

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

Roger Levy and Christopher D. Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 439–446, Sapporo, Japan, July. Association for Computational Linguistics.

Junhui Li, Guodong Zhou, and Hwee Tou Ng. 2010. Joint syntactic and semantic parsing of chinese. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1108–1117.

Yang Liu and Qun Liu. 2010. Joint parsing and translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 707–715.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL 2006*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, pages 91–98.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. In *Computational Linguistics*, volume 34, pages 513–553.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL 2007*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP 1996*.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague, Czech Republic, June. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL-2008*.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 486–494.

Xinyan Xiao, Yang Liu, YoungSook Hwang, Qun Liu, and Shouxun Lin. 2010. Joint tokenization and translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1200–1208.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*, volume 11, pages 207–238.

Yue Zhang and Stephen Clark. 2008a. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896.

Yue Zhang and Stephen Clark. 2008b. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii, October. Association for Computational Linguistics.

Relaxed Cross-lingual Projection of Constituent Syntax

Wenbin Jiang and **Qun Liu** and **Yajuan Lü**

Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{jiangwenbin, liuqun, lvyajuan}@ict.ac.cn

Abstract

We propose a relaxed correspondence assumption for cross-lingual projection of constituent syntax, which allows a supposed constituent of the target sentence to correspond to an unrestricted treelet in the source parse. Such a relaxed assumption fundamentally tolerates the syntactic non-isomorphism between languages, and enables us to learn the target-language-specific syntactic idiosyncrasy rather than a strained grammar directly projected from the source language syntax. Based on this assumption, a novel constituency projection method is also proposed in order to induce a projected constituent treebank from the source-parsed bilingual corpus. Experiments show that, the parser trained on the projected treebank dramatically outperforms previous projected and unsupervised parsers.

1 Introduction

For languages with treebanks, supervised models give the state-of-the-art performance in dependency parsing (McDonald and Pereira, 2006; Nivre et al., 2006; Koo and Collins, 2010; Martins et al., 2010) and constituent parsing (Collins, 2003; Charniak and Johnson, 2005; Petrov et al., 2006). To break the restriction of the treebank scale, lots of works have been devoted to the unsupervised methods (Klein and Manning, 2004; Bod, 2006; Seginer, 2007; Cohen and Smith, 2009) and the semi-supervised methods (Sarkar, 2001; Steedman et al., 2003; McClosky et al., 2006; Koo et al., 2008) to utilize the unannotated text. In recent years, researchers have also conducted many investigations on syntax projection (Hwa et al., 2005; Ganchev et al., 2009; Smith and Eisner, 2009; Jiang et al., 2010), in order to borrow syntactic knowledge from another language.

Different from the bilingual parsing (Smith and Smith, 2004; Burkett and Klein, 2008; Zhao et al., 2009; Huang et al., 2009; Chen et al., 2010) that improves parsing performance with bilingual constraints, and the bilingual grammar induction (Wu, 1997; Kuhn, 2004; Blunsom et al., 2008; Snyder et al., 2009) that induces grammar from parallel text, the syntax projection aims to project the syntactic knowledge from one language to another. This seems especially promising for the languages that have bilingual corpora parallel to resource-rich languages with large treebanks. Previous works mainly focus on dependency projection. The dependency relationship between words in the parsed source sentences can be directly projected across the word alignment to words in the target sentences, following the direct correspondence assumption (DCA) (Hwa et al., 2005). Due to the syntactic non-isomorphism between languages, DCA assumption usually leads to conflicting or incomplete projection. Researchers have to adopt strategies to tackle this problem, such as designing rules to handle language non-isomorphism (Hwa et al., 2005), and resorting to the quasi-synchronous grammar (Smith and Eisner, 2009).

For constituency projection, however, the lack of isomorphism becomes much more serious, since a constituent grammar describes a language in a more detailed way. In this paper we propose a relaxed correspondence assumption (RCA) for constituency

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1192–1201,
Edinburgh, Scotland, UK, July 27–31, 2011.

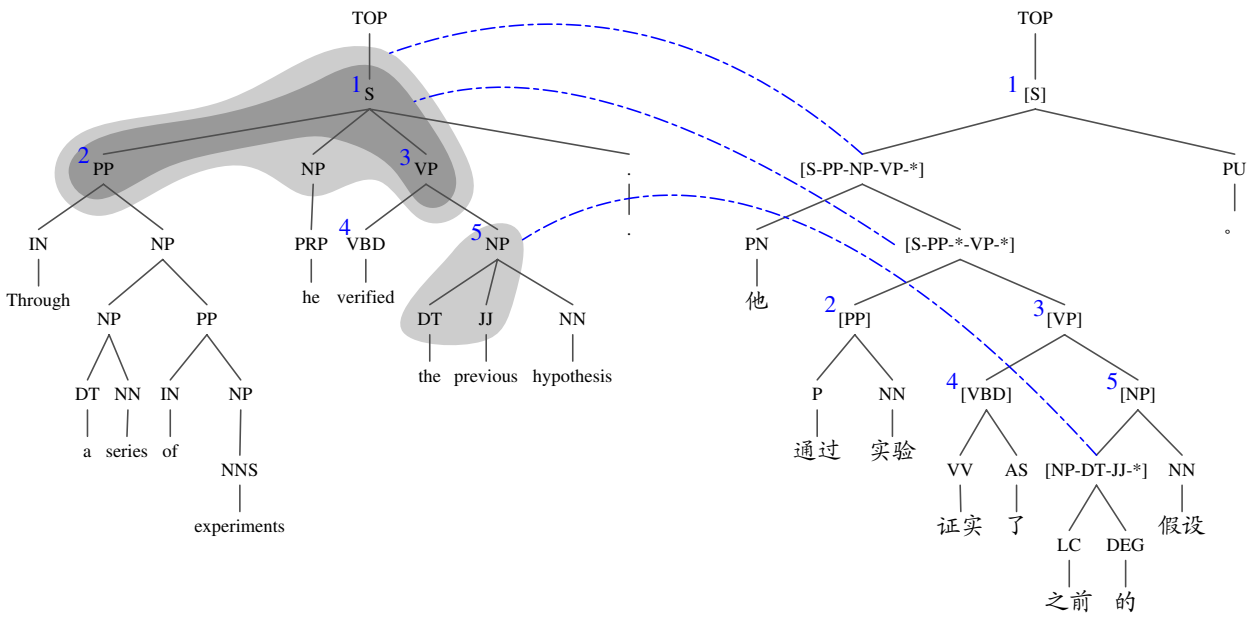


Figure 1: An example for constituency projection based on the RCA assumption. The projection is from English to Chinese. A dash dot line links a projected constituent to its corresponding treelet, which is marked with gray background; An Arabic numeral relates a directly-projected constituent to its counter-part in the source parse.

projection. It allows a supposed constituent of the target sentence to correspond to an unrestricted treelet in the source parse. Such a relaxed assumption fundamentally tolerates the syntactic non-isomorphism between languages, and enables us to learn the target-language-specific syntactic idiosyncrasy, rather than induce a strained grammar directly projected from the source language syntax. We also propose a novel cross-lingual projection method for constituent syntax based on the RCA assumption. Given a word-aligned source-parsed bilingual corpus, a PCFG grammar can be induced for the target language by maximum likelihood estimation on the exhaustive enumeration of candidate projected productions, where each nonterminal in a production is an unrestricted treelet extracted from the source parse. The projected PCFG grammar is then used to parse each target sentence under the guidance of the corresponding source tree, so as to produce an optimized projected constituent tree.

Experiments validate the effectiveness of the RCA assumption and the constituency projection method. We induce a projected Chinese constituent treebank from the FBIS Chinese-English parallel corpus with English sentences parsed by the Charniak parser. The Berkeley Parser trained on the projected treebank dramatically outperforms the previous projected and unsupervised parsers. This provides an promising substitute for unsupervised parsing methods, to the resource-scarce languages that have bilingual corpora parallel to resource-rich languages with human-annotated treebanks.

In the rest of this paper we first presents the RCA assumption, and the algorithm used to determine the corresponding treelet in the source parse for a candidate constituent in the target sentence. Then we describe the induction of the projected PCFG grammar and the projected constituent treebank from the word-aligned source-parsed parallel corpus. After giving experimental results and the comparison with previous unsupervised and projected parsers, we finally conclude our work and point out several aspects to be improved in the future work.

2 Relaxed Correspondence Assumption

The DCA assumption (Hwa et al., 2005) works well in dependency projection. A dependency grammar describes a sentence in a compact manner where the syntactic information is carried by the dependency relationships between pairs of words. It is reasonable to audaciously assume that the relationship of

Algorithm 1 Treelet Extraction Algorithm.

1: **Input**: $\mathbf{T_f}$: parse tree of source sentence $\mathbf{f}$
2: $\mathbf{e}$: target sentence
3: $\mathbf{A}$: word alignment of $\mathbf{e}$ and $\mathbf{f}$
4: **for** i, j s.t. $1 \leq i < j \leq |\mathbf{e}|$ **do** ▷ all spans
5: $\mathbf{t} \leftarrow \text{EXTTREELET}(\mathbf{e}, i, j, \mathbf{T_f}, \mathbf{A})$
6: $\mathbb{T}_{\langle i,j \rangle} \leftarrow \text{PRUNETREE}(\mathbf{t})$
7: **Output:** treelet set $\mathbb{T}$ for all spans of $\mathbf{e}$
8: **function** EXTTREELET($\mathbf{e}$, i, j, $\mathbf{T}$, $\mathbf{A}$)
9: **if** $\mathbf{T}$ aligns totally outside $\mathbf{e}_{i:j}$ **then**
10: **return** $\emptyset$
11: **if** $\mathbf{T}$ aligns totally inside $\mathbf{e}_{i:j}$ **then**
12: **return** $\{\mathbf{T} \cdot root\}$
13: $\mathbf{t} \leftarrow \{\mathbf{T} \cdot root\}$ ▷ partly aligned inside $\mathbf{e}_{i:j}$
14: **for** each subtree $\mathbf{s}$ of $\mathbf{T}$ **do**
15: $\mathbf{t} \leftarrow \mathbf{t} \cup \text{EXTTREELET}(\mathbf{e}, i, j, \mathbf{s}, \mathbf{A})$
16: **return** $\mathbf{t}$
17: **function** PRUNETREE($\mathbf{T}$)
18: **for** each node n in $\mathbf{T}$ **do**
19: merge n's successive empty children
20: $\mathbf{t} \leftarrow \mathbf{T}$
21: **while** $\mathbf{t}$ has only one non-empty subtree **do**
22: $\mathbf{t} \leftarrow$ the non-empty subtree of $\mathbf{t}$
23: **return** $\mathbf{t}$

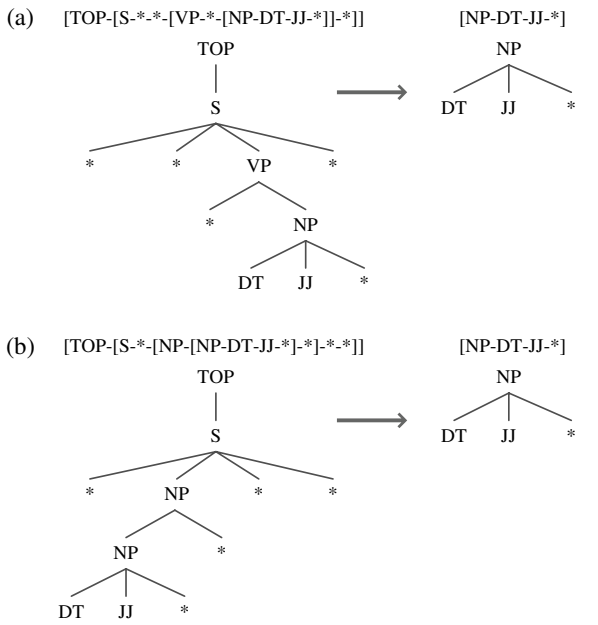


Figure 2: Two examples for treelet pruning. Asterisks indicate eliminated subtrees, which are represented as empty children of their parent nodes.

a word pair in the source sentence also holds for the corresponding word pair in the target sentence. Compared with dependency grammar, constituent grammar depicts syntax in a more complex way that gives a sentence a hierarchically branched structure. Therefore the lack of syntactic isomorphism for constituency projection becomes much more serious, it will be hard and inappropriate to directly project the complex constituent structure from one language to another.

For constituency projection, we propose a relaxed corresponding assumption (RCA) to eliminate the influence of syntactic non-isomorphism between the source- and target languages. This assumption allows a supposed constituent of the target sentence to correspond to an unrestricted treelet in the source parse. A treelet is a connected subgraph in the source constituent tree, which covers a discontiguous sequence of words of the source sentence. This property enables a supposed constituent of the target sentence not necessarily to correspond to exactly a constituent of the source parse, so as to fundamentally tolerate the syntactic non-isomorphism between languages. Figure 1 gives an example of relaxed correspondence.

2.1 Corresponding Treelet Extraction

According to the word alignment between the source and target sentences, we can extract the treelet out of the source parse for any possible constituent span of the target sentence. Algorithm 1 shows the treelet extraction algorithm.

Given the target sentence $\mathbf{e}$, the parse tree $\mathbf{T_f}$ of the source sentence $\mathbf{f}$, and the word alignment $\mathbf{A}$ between $\mathbf{e}$ and $\mathbf{f}$, the algorithm extracts the corresponding treelet out of $\mathbf{T_f}$ for each candidate span of $\mathbf{e}$ (line 4-6). For a given span $\langle i, j \rangle$, its corresponding treelet in $\mathbf{T_f}$ can be extracted by a recursive top-down traversal in the tree. If all nodes in the current subtree $\mathbf{T}$ align outside of source subsequence $\mathbf{e}_{i:j}$, the recursion stops and returns an empty tree $\emptyset$, indicating that the subtree is eliminated from the final treelet (line 9-10). And, if all nodes in $\mathbf{T}$ align inside $\mathbf{e}_{i:j}$, the root of $\mathbf{T}$ is returned as the concise representation of the whole subtree (line 11-12). For the third situation, that is to say $\mathbf{T}$ aligns partly inside $\mathbf{e}_{i:j}$, the recursion has to continue to investigate the subtrees of $\mathbf{T}$ (line 14-15). The recursive traversal finally returns a treelet $\mathbf{t}$ that exactly corre-

sponds to the candidate constituent span $\langle i, j \rangle$ of the source sentence.

We can find that even for a smaller span, the recursive extraction procedure still starts from the root of the source tree. This leads to a expatiatory treelet with some redundant nodes on the top. Function PRUNETREE takes charge of the treelet pruning (line 6). It traverses the treelet to merge the successive empty sibling nodes (marked with asterisks) into one (line 18-19), then conducts a top-down pruning to delete the redundant branches until meeting a branch with more than one non-empty subtrees (line 20-22). Figure 2 shows the effect of the pruning operation with two examples. The pruning operation maps the two original treelets into the same simplified version, that is, the pruned treelet. The branches pruned out of the original treelet serve as the context of the pruned treelet. The bracketed representations of the pruned treelets, as shown above the treelet graphs, are used as the nonterminals of the projected target parses.

Since the overall complexity of the algorithm is $O(|\mathbf{e}|^3)$, it seems inefficient to collect the treelets for all spans in the target sentence. But in fact it runs fast on the realistic corpus in our experiments, we assume that the function EXTTREELET doesn't always consume $O(|\mathbf{e}|)$ because of the more or less isomorphism between two languages.

3 Projected Grammar and Treebank

This section describes how to build a projected constituent treebank based on the RCA assumption. According to the last section, each span of the target sentence could correspond to a treelet in the source parse. If a span $\langle i, j \rangle$ has a corresponding treelet $\mathbf{t}$, a candidate projected constituent can be defined as a triple $\langle i, j, \mathbf{t} \rangle$. For an n-way partition of this span,

$$\langle i, k_1 \rangle, \langle k_1 + 1, k_2 \rangle, .., \langle k_{n-1} + 1, j \rangle$$

if each sub-span $\langle k_{p-1}+1, k_p \rangle$ corresponds to a candidate constituent $\langle k_{p-1}+1, k_p, \mathbf{t}_p \rangle$, a candidate projected production can then be defined, denoted as

$$\langle i, j, \mathbf{t} \rangle \rightarrow \langle i, k_1, \mathbf{t}_1 \rangle \langle k_1+1, k_2, \mathbf{t}_2 \rangle .. \langle k_{n-1}+1, j, \mathbf{t}_n \rangle$$

There may be many candidate projected constituents because of arbitrary combination, the tree projection procedure aims to find the optimum tree from the parse forest determined by these candidate constituents. Each production in the optimum tree should satisfy this principle: the rule used in this production appears in the whole corpus as frequently as possible.

However, due to translation diversity and word alignment error, the real constituent tree of the target sentence may not be contained in the candidate projected constituents. We propose a relaxed and fault-tolerant tree projection strategy to tackle this problem. First, based on the distribution of candidate projected constituents over each single sentence, we estimate the distribution over the whole corpus for the rules used in these constituents, so as to obtain a projected PCFG grammar. Then, using a PCFG parser and this grammar, we parse each target sentence under the guidance of the candidate projected constituent set of the target sentence, so as to obtain the optimum projected tree as far as possible. In the following, we first describe the estimation of the projected PCFG grammar and then show the tree projection procedure.

3.1 Projected PCFG Grammar

From a human-annotated treebank, we can induce a PCFG grammar by estimating the frequency of the production rules, which are contained in the productions of the trees. But for each target sentence we don't know which candidate productions consist the correct constituent tree, so we can't estimate the frequency of the production rules directly.

A reasonable hypothesis is, if a candidate projected production for a target sentence happens to be in the correct parse of the sentence, the rule used in this production will appear frequently in the whole corpus. We assume that each candidate projected production may be a part of the correct parse, but with different probabilities. If we give each candidate projected production an appropriate probability and use this probability as the appearance frequency of this production in the correct parse, we can achieve an approximation of the PCFG grammar hidden in the target sentences. In this work, we restrict the productions to be binarized to reduce the computational complexity. It results in a binarized PCFG grammar, similar to previous unsupervised works.

To estimate the frequencies of the candidate pro-

ductions in the correct parse of the target sentence, we need first estimate the frequencies of the candidate spans, which are described as follows:

$$p(\langle i,j\rangle|\mathbf{e}) = \frac{\#\ of\ trees\ including\ \langle i,j\rangle}{\#\ of\ all\ trees} \quad (1)$$

The count of all binary trees of a target sentence $\mathbf{e}$ can be calculated similar to the β value calculation in the inside-outside algorithm. Without confusion, we adopt the symbol $\beta(i,j)$ to denote the count of binary tree for span $\langle i,j\rangle$:

$$\beta(i,j) = \begin{cases} 1 & i = j \\ \sum_{k=i}^{j-1} \beta(i,k) \cdot \beta(k+1,j) & i < j \end{cases} \quad (2)$$

$\beta(1,|\mathbf{e}|)$ is the count of binary trees of target sentence $\mathbf{e}$. We also need to calculate the count of binary tree fragments that cover the nodes outside span $\langle i,j\rangle$. This is similar to the calculation of the α value in the inside-outside algorithm. We also adopt the symbol $\alpha(i,j)$ here:

$$\alpha(i,j) = \begin{cases} 1 & i = 1, j = |\mathbf{e}| \\ \sum_{k=j+1}^{|\mathbf{e}|} \alpha(i,k) \cdot \beta(k+1,|\mathbf{e}|) & \\ + \sum_{k=1}^{i-1} \alpha(k,j) \cdot \beta(k,j-1) & \mathbf{else} \end{cases} \quad (3)$$

For simplicity we omit some conditions in above formulas. The count of trees containing span $\langle i,j\rangle$ is $\alpha(i,j) \cdot \beta(i,j)$. Equation 1 can be rewritten as

$$p(\langle i,j\rangle|\mathbf{e}) = \frac{\alpha(i,j) \cdot \beta(i,j)}{\beta(1,|\mathbf{e}|)} \quad (4)$$

On condition that $\langle i,j\rangle$ is a span in the parse of $\mathbf{e}$, the probability that $\langle i,j\rangle$ has two children $\langle i,k\rangle$ and $\langle k+1,j\rangle$ is

$$p(\langle i,k\rangle\langle k+1,j\rangle|\langle i,j\rangle) = \frac{\beta(i,k) \cdot \beta(k+1,j)}{\beta(i,j)} \quad (5)$$

Therefore, the probability that $\langle i,j\rangle$ is a span in the parse of $\mathbf{e}$ and has two children $\langle i,k\rangle$ and $\langle k+1,j\rangle$ can be calculated as follows:

$$\begin{aligned} &p(\langle i,j\rangle \rightarrow \langle i,k\rangle\langle k+1,j\rangle|\mathbf{e}) \\ &\quad = p(\langle i,j\rangle|\mathbf{e}) \cdot p(\langle i,k\rangle\langle k+1,j\rangle|\langle i,j\rangle) \\ &\quad = \frac{\alpha(i,j) \cdot \beta(i,k) \cdot \beta(k+1,j)}{\beta(1,|\mathbf{e}|)} \end{aligned} \quad (6)$$

Since each candidate projected span aligns to one treelet at most, this probability is also the frequency of the candidate projected production related to the three spans.

The counting approach above is based on the assumption that there is a uniform distribution over the projected trees for every target sentence. The inside and outside algorithms and the other counting formulae are used to calculate the expected counts under this assumption. This looks like a single iteration of EM.

A binarized projected PCFG grammar can then be easily induced by maximum likelihood estimation. Due to word alignment errors, free translation, and exhaustive enumeration of possible projected productions, such a PCFG grammar may contain too much noisy nonterminals and production rules. We introduce a threshold b_{RULE} to filter the grammar. A production rule can be reserved only if its frequency is larger than b_{RULE}.

3.2 Relaxed Tree Projection

The projected PCFG grammar is used in the procedure of constituency projection. Such a grammar, as a kind of global syntactic knowledge, can attenuate the negative effect of word alignment error, free translation and syntactic non-isomorphism for the constituency projection between each single sentence pair. To obtain as optimal a projected constituency tree as possible, we have to integrate two kinds of knowledge: the local knowledge in the candidate projected production set of the target sentence, and the global knowledge in the projected PCFG grammar.

The integrated projection strategy can be conducted as follows. We parse each target sentence with the projected PCFG grammar $\mathbf{G}$, and use the candidate projected production set $\mathbf{D}$ to guide the PCFG parsing. The parsing procedure aims to find an optimum projected tree, which maximizes both the PCFG tree probability and the count of productions that also appear in the candidate projected pro-

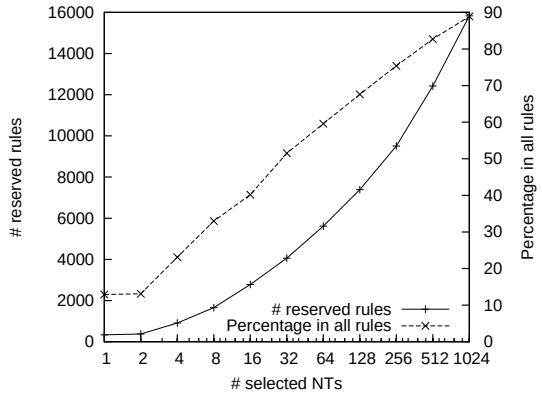


Figure 3: Rule counts corresponding to selected nonterminal sets, and their frequency summation proportions to the whole rule set.

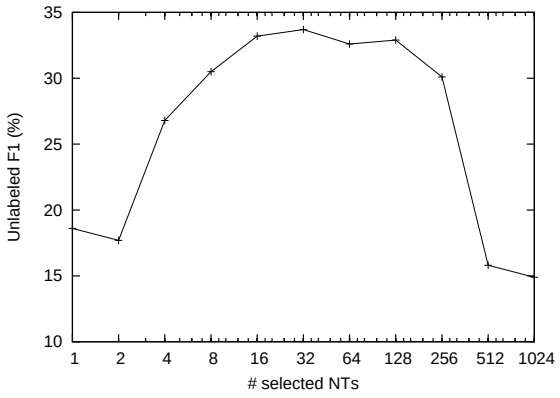


Figure 4: Performance curve of the projected PCFG grammars corresponding to different sizes of nonterminal sets.

duction set. The two optimization objectives can be coordinated as follows:

$$\tilde{y} = \underset{y}{\operatorname{argmax}} \prod_{d \in y} (p(d|\mathbf{G}) \cdot e^{\lambda \cdot \delta(d, \mathbf{D})}) \quad (7)$$

Here, d represents a production; δ is a boolean function that returns 1 if d appears in $\mathbf{D}$ and returns 0 otherwise; λ is a weight coefficient that needs to be tuned to maximize the quality of the projected treebank.

4 Experiments

Our work focuses on the constituency projection from English to Chinese. The FBIS Chinese-English parallel corpus is used to obtain a projected constituent treebank. It contains 239 thousand sentence pairs, with about 6.9/8.9 million Chinese/English words. We parse the English sentences with the Charniak Parser (Charniak and Johnson, 2005), and tag the Chinese sentences with a POS tagger implemented faithfully according to (Collins, 2002) and trained on the Penn Chinese Treebank 5.0 (Xue et al., 2005). We perform word alignment by runing GIZA++ (Och and Ney, 2000), and then use the alignment results for constituency projection.

Following the previous works of unsupervised constituent parsing, we evaluate the projected parser on the subsets of CTB 1.0 and CTB 5.0, which contain no more than 10 or 40 words after the removal of punctuation. The gold-standard POS tags are directly used for testing. The evaluation for unsupervised parsing differs slightly from the standard PARSEVAL metrics, it ignores the multiplicity of brackets, brackets of span one, and the bracket labels. In all experiments we report the unlabeled F1 value which is the harmonic mean of the unlabeled precision and recall.

4.1 Projected PCFG Grammar

An initial projected PCFG grammar can be induced from the word-aligned and source-parsed parallel corpus according to section 3.1. Such an initial grammar is huge and contains a large amount of projected nonterminals and production rules, where many of them come from free translation and word alignment errors. We conservatively set the filtration threshold b_{RULE} as 1.0 to discard the rules with frequency less than one, the rule count falls dramatically from 3.3 millions to 92 thousands.

Figure 3 shows the statistics of the remained production rules. We sort the projected nonterminals according to their frequencies and select the top 2^N ($1 \leq N \leq 10$) best ones, and then discard the rules that fall out of the selected nonterminal set. The frequency summation of the rule set corresponding to 32 best nonterminals accounts for nearly 90% of the frequency summation of the whole rule set.

We use the developing set of CTB 1.0 (chapter 301-325) to evaluate the performance of a series of filtered grammars. Figure 4 gives the unlabeled F1 value of each grammar on all trees in the developing set. The filtered grammar corresponding to the set of top 32 nonterminals achieves the highest performance. We denote this grammar as $\mathbf{G}_{32}$ and use it

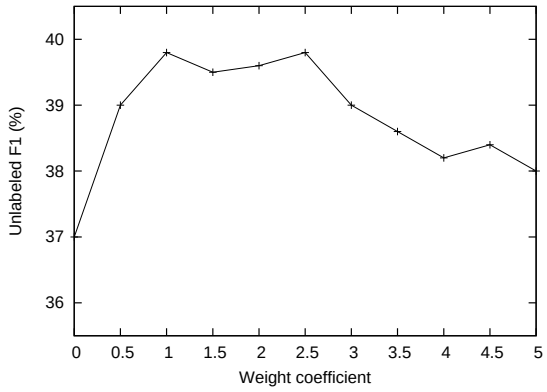


Figure 5: Performance curve of the Berkeley Parser trained on 5 thousand projected trees. The weight coefficient λ ranges from 0 to 5.

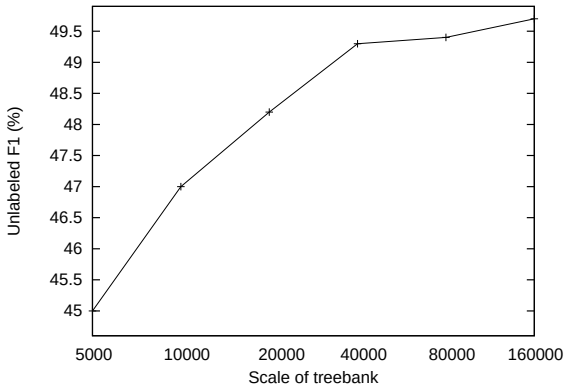


Figure 6: Performance curve of the Berkeley Parser trained on different amounts of best project trees. The scale of the selected treebank ranges from 5000 to 160000.

in the following tree projection procedure.

4.2 Projected Treebank and Parser

The projected grammar $\mathbf{G}_{32}$ provides global syntactic knowledge for constituency projection. Such global knowledge and the local knowledge carried by the candidate projected production set are integrated in a linear weighted manner as in Formula 7. The weight coefficient λ is tuned to maximize the quality of the projected treebank, which is indirectly measured by evaluating the performance of the parser trained on it.

We select the first 5 thousand sentence pairs from the Chinese-English FBIS corpus, and induce a series of projected treebanks using different λ, ranging from 0 to 5. Then we train the Berkeley Parser on each projected treebank, and test it on the developing set of CTB 1.0. Figure 5 gives the performance curve, which reports the unlabeled F1 values of the projected parsers on all sentences of the developing set. We find that the best performance is achieved with λ between 1 and 2.5, with slight fluctuation in this range. It can be concluded that, the projected PCFG grammar and the candidate projected production set do represent two different kinds of constraints, and we can effectively coordinate them by tuning the weight coefficient. Since different λ values in this range result in slight performance fluctuation of the projected parser, we simply set it to 1 for the constituency projection on the whole FBIS corpus.

There are more than 200 thousand projected trees induced from the Chinese-English FBIS corpus. It is a heavy burden for a parser to train on so large a treebank. And on the other hand, the free translation and word alignment errors result in many projected trees of poor-quality. We design a criteria to approximate the quality of the projected tree y for the target sentence x:

$$\tilde{Q}(y) = \sqrt[|x|-1]{\prod_{d \in y} (p(d|\mathbf{G}) \cdot e^{\lambda \cdot \delta(d, \mathbf{D})})} \qquad (8)$$

and use an amount of best projected trees instead of the whole projected treebank to train the parser. Figure 6 shows the performance of the Berkeley Parser trained on different amounts of selected trees. The performance of the Berkeley Parser constantly improves along with the increment of selected trees. However, treebanks containing more than 40 thousand projected trees can not brings significant improvement. The parser trained on 160 thousand trees only achieves an F1 increment of 0.4 points over the one trained on 40 thousand trees. This indicates that the newly added trees do not give the parser more information due to their projection quality, and a larger parallel corpus may lead to better parsing performance.

The Berkeley Parser trained on 160 thousand best projected trees is used in the final test. Table 1 gives the experimental results and the comparison with related works. This is a sparse table since the experiments of previous researchers focused on different data sets. Our projected parser significantly

System	CTB-TEST-40	CTB1-ALL-10	CTB5-ALL-10	CTB5-ALL-40
(Klein and Manning, 2004)	—	46.7	—	—
(Bod, 2006)	—	47.2	—	—
(Seginer, 2007)	—	—	**54.6**	38.0
(Jiang et al., 2010)	40.4	—	—	—
our work	**52.1**	**54.4**	**54.5**	**49.2**

Table 1: The performance of the Berkeley Parser trained on 160 thousand best projected trees, compared with previous works on constituency projection and unsupervised parsing. CTB-TEST-40: sentences ≤ 40 words from CTB standard test set (chapter 271-300); CTB1-ALL-10/CTB5-ALL-10: sentences ≤ 10 words from CTB 1.0/CTB 5.0 after the removal of punctuation; CTB5-ALL-40: sentences ≤ 40 words from CTB 5.0 after the removal of punctuation.

outperforms the parser of Jiang et al. (2010), where they directly adapt the DCA assumption of (Hwa et al., 2005) from dependency projection to constituency projection and resort to a better word alignment and a more complicated tree projection algorithm. This indicates that the RCA assumption is more suitable for constituency projection than the DCA assumption, and can induce a better grammar that much more reflects the language-specific syntactic idiosyncrasy of the target language.

Our projected parser also obviously surpasses existing unsupervised parsers. The parser of Seginer (2007) performs slightly better on CTB 5.0 sentences no more than 10 words, but obviously falls behind on sentences no more than 40 words. Figure 7 shows the unlabeled F1 of our parser on a series of subsets of CTB 5.0 with different sentence length upper limits. We find that even on the whole treebank, our parser still gives a promising result. Compared with unsupervised parsing, constituency projection can make use of the syntactic information of another language, so that it probably induce a better grammar. Although comparing a syntax projection technique to supervised or semi-supervised techniques seems unfair, it still suggests that if a resource-poor language has a bilingual corpus parallel to a resource-rich language with a human-annotated treebank, the constituency projection based on RCA assumption is a promising substitute for unsupervised parsing.

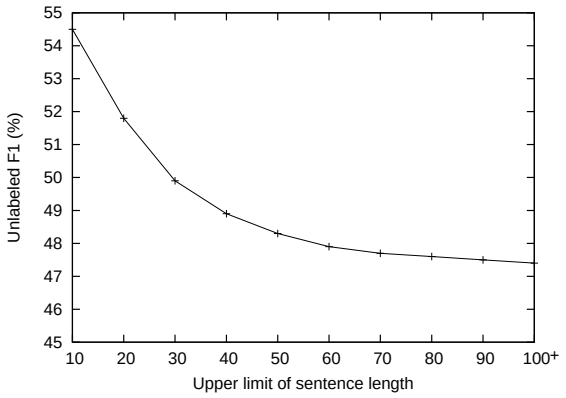


Figure 7: Performance of the Berkeley Parser on subsets of CTB 5.0 with different sentence length upper limits. 100+ indicates the whole treebank.

5 Conclusion and Future Works

This paper describes a relaxed correspondence assumption (RCA) for constituency projection. Under this assumption a supposed constituent in the target sentence can correspond to an unrestricted treelet in the parse of the source sentence. Different from the direct correspondence assumption (DCA) widely used in dependency projection, the RCA assumption is more suitable for constituency projection, since it fundamentally tolerates the syntactic non-isomorphism between the source and target languages. According to the RCA assumption we propose a novel constituency projection method. First, a projected PCFG grammar is induced from the word-aligned source-parsed parallel corpus. Then, the tree projection is conducted on each sentence pair by a PCFG parsing procedure, which integrates both the global knowledge in the projected PCFG grammar and the local knowledge in the set of candidate projected productions.

Experiments show that the parser trained on the projected treebank significantly outperforms the projected parsers based on the DCA assumption. This validates the effectiveness of the RCA assumption and the constituency projection method, and indicates that the RCA assumption is more suit-

able for constituency projection than the DCA assumption. The projected parser also obviously surpasses the unsupervised parsers. This suggests that if a resource-poor language has a bilingual corpus parallel to a resource-rich language with a human-annotated treebank, the constituency projection based on RCA assumption is an promising substitute for unsupervised methods.

Although achieving appealing results, our current work is quite coarse and has many aspects to be improved. First, the word alignment is the fundamental precondition for projected grammar induction and the following constituency projection, we can adopt the better word alignment strategies to improve the word alignment quality. Second, the PCFG grammar is too weak due to its context free assumption, we can adopt more complicated grammars such as TAG (Joshi et al., 1975), in order to provide a more powerful global syntactic constraints for the tree projection procedure. Third, the current tree projection algorithm is too simple, more bilingual constraints could lead to better projected trees. Last but not least, the constituency projection and the unsupervised parsing make use of different kinds of knowledge, therefore the unsupervised methods can be integrated into the constituency projection framework to achieve better projected grammars, treebanks, and parsers.

Acknowledgments

The authors were supported by National Natural Science Foundation of China Contract 90920004, 60736014 and 60873167. We are grateful to the anonymous reviewers for their thorough reviewing and valuable suggestions.

References

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. Bayesian synchronous grammar induction. In *Proceedings of the NIPS*.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the COLING-ACL*.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of the EMNLP*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine-grained n-best parsing and discriminative reranking. In *Proceedings of the ACL*.

Wenliang Chen, Jun' ichi Kazama, and Kentaro Torisawa. 2010. Bitext dependency parsing with bilingual subtree constraints. In *Proceedings of the ACL*.

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of the NAACL-HLT*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the 47th ACL*.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of the EMNLP*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. In *Natural Language Engineering*.

Wenbin Jiang, Yajuan Lü, Yang Liu, and Qun Liu. 2010. Effective constituent projection across languages. In *Proceedings of the COLING*.

A. K. Joshi, L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal Computer Systems Science*.

Dan Klein and Christopher D. Manning. 2004. Corpusbased induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the ACL*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the ACL*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the ACL*.

Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Proceedings of the ACL*.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of EMNLP*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the ACL*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Joakim Nivre, Johan Hall, Jens Nilsson, Gulsen Eryigit, and Svetoslav Marinov. 2006. Labeled pseudoprojective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the ACL*.

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

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of NAACL*.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the ACL*.

David Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of EMNLP*.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *Proceedings of the EMNLP*.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the ACL*.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the EACL*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of the ACL-IJCNLP*.

Computing Logical Form on Regulatory Texts*

Nikhil Dinesh
Artificial Intelligence Center
SRI International
Menlo Park, CA - 94025
dinesh@ai.sri.com

Aravind Joshi and **Insup Lee**
Department of Computer Science
University of Pennsylvania
Philadelphia, PA - 19104
{joshi,lee}@seas.upenn.edu

Abstract

The computation of logical form has been proposed as an intermediate step in the translation of sentences to logic. Logical form encodes the resolution of scope ambiguities. In this paper, we describe experiments on a modest-sized corpus of regulation annotated with a novel variant of logical form, called *abstract syntax trees* (ASTs). The main step in computing ASTs is to order scope-taking operators. A learning model for ranking is adapted for this ordering. We design features by studying the problem of comparing the scope of one operator to another. The scope comparisons are used to compute ASTs, with an F-score of 90.6% on the set of ordering decisons.

1 Introduction

May (1985) argued for a level of *logical form* as a prelude to translating sentences to logic. Just as a parse tree determines the constituent structure of a sentence, a logical form of a sentence represents one way of resolving scope ambiguities. The level of logical form is an appealing layer of modularity; it allows us to take a step beyond parsing in studying scope phenomenon, and yet, avoid the open problem of fully translating sentences to logic.

Data-driven analyses of scope have been of interest in psycholinguistics (Kurtzman and MacDonald, 1993) and more recently in NLP (Srinivasan and Yates, 2009). The focus has typically been on predicting the preferred scopal ordering of sentences with two quantifying determiners, for example, in the sentence "every kid climbed a tree". In the related problem of translating database queries to logic, Zettlemoyer and Collins (2009) and Wong and Mooney (2007) consider the scope of adjectives in addition to determiners, for example the scope of "cheapest" in the noun phrase "the cheapest flights from Boston to New York". To our knowledge, empirical studies of scope have been restricted to phenomenon between and within noun phrases.

In this paper, we describe experiments on a novel annotation of scope phenomenon in regulatory texts – Section 610 of the Food and Drug Administration's Code of Federal Regulations[1] (FDA CFR). Determiners, modals, negation, and verb phrase modifiers are the main scope-taking operators. We have annotated 195 sentences with a variant of logical form, called *abstract syntax trees* (ASTs). Our focus is on the problem of computing the AST, given a (variant of a) parse tree of a sentence.

The long term goal of this work is to assist in the translation of regulation to logic, for the application of conformance checking. The problem is to formally determine whether an organization conforms to regulation, by checking the organization's records using the logical translation of regulation. Conformance checking has been of interest in a variety of regulatory contexts, and examples include privacy policy (Barth et al., 2006; Jones and Sergot, 1992; Anderson, 1996) and business contracts (Governatori et al., 2006; Grosof et al., 1999).

We now discuss some problems that arise in defin-

*This research was supported in part by ONR MURI N00014-07-1-0907, NSF CNS-1035715, NSF IIS 07-05671, and SRI International.

[1] http://www.gpoaccess.gov/cfr/index.html

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1202–1212,
Edinburgh, Scotland, UK, July 27–31, 2011.

ing logical form and the assumptions that we make to circumvent these problems.

1.1 Problems and Assumptions

A key assumption of logical form is that the translation from language to logic is syntax-based. As a result, the logic needs to be expressive enough to accomodate a syntactic translation. There is no consensus logic for constructs, such as, plurals, purpose clauses, and certain modals. This leads to the following problem in defining logical form.

How do we define the logical form of a sentence, without defining the logic? We adopt a specific formalism that accomodates a subset of the constructs found in regulation. We generalize from the formalized constructs to other constructs. Some of these generalizations may need revision in the future.

We assume that sentences in regulation are translated to statements in logic of the form:

$$(\mathrm{id})\ \varphi(x_1, ..., x_n) \mapsto \psi(x_1, ..., x_n)$$

where, "id" is an identifier, φ is *the precondition*, ψ is *the postcondition*, and $x_1, ..., x_n$ are free variables. The distinction between pre and postconditions has been adopted by most logics for regulation, to accomodate exceptions to laws (Sergot et al., 1986; Makinson and van der Torre, 2000; Governatori et al., 2006). The pre and postconditions are expressed in a modal logic that we designed in prior work (Dinesh et al., 2011). In describing the logical form, we will sketch how the logical form can be mapped to logic. But, we do not assume that the reader has a detailed understanding of the logic.

Given the assumptions about the logic, our goal is to transform a regulatory sentence into a structure that lets us determine: (I) the constituents of a sentence that contribute to the pre/postcondition, and (II) the scope of operators in the pre/postcondition. The structures that we use are called *abstract syntax trees* (ASTs), which can be understood as a restricted kind of logical form for regulatory texts.

1.2 Contributions and Outline

In this paper, we focus on the problem of computing the AST given a (kind of) parse tree for a sentence. The main step is is to *order or rank* scope-taking operators. A learning model for ranking is adapted for this ordering. We design features by studying the problem of comparing the scope of one operator to another. The pairwise scope comparisons are then used to compute ASTs, with an F-score of 90.6% on the set of ordering decisons.

The rest of this paper is organized as follows. We define ASTs using an example in Section 2, and setup the learning problem in Section 3. We then describe the corpus using statistics about operators in Section 4. In Section 5, we describe experiments on comparing the scope of an operator to another. We use the pairwise scope comparisons, in Section 6 to comput the AST. We discuss related work in Section 7 and conclude in Section 8.

2 Abstract Syntax Trees

We describe *abstract syntax trees* (ASTs) using an example from CFR Section 610.11:

(1) A general safety test for the detection of extraneous toxic contaminants shall be performed on biological products intended for administration to humans.

We discuss the translation in logic and the AST for the fragment of (1) that appears in black. In order to keep figures to a manageable size, we restrict attention to fragments of sentences, by graying out portions. The term AST is borrowed from compilers (Aho et al., 1986), where it is used as an intermediate step in the semantic interpretation of programs.

Translation in Logic: The sentence (1) is formally expressed as:

$$(1)\ \mathrm{bio\_prod}(x) \mapsto \mathcal{O}_{m(x)}(\exists y : \mathrm{test}(y) \wedge \psi(x, y))$$

where, $\psi(x, y) = \mathrm{gensaf}(y) \wedge \mathrm{ag}(y, m(x)) \wedge \mathrm{ob}(y, x)$

The predicates and function symbols are read as follows. $\mathrm{bio\_prod}(x)$ - "x is a biological product". $m(x)$ denotes the manufacturer of x. The modal operator $\mathcal{O}$ stands for "obligation". $\mathrm{test(y)}$ - "y is a test (event)". $\mathrm{gensaf}(y)$ - "y is a general safety procedure". $\mathrm{ag}(y, m(x))$ - "the agent of y is $m(x)$", and $\mathrm{ob}(y, x)$ - "the object of the event y is x". The formalized version of the law is read as follows: "If x is a biological product, then the manufacturer $m(x)$ is required/obligated to perform a general safety test y which has x as its object". We refer the reader to (Dinesh et al., 2011) for details on the logic.

The distinction between pre and postconditions is a non-trivial assumption. As with all logic-programming formalisms, only free variables are "shared" between pre and postconditons. This implies that all existential quantification, modals, and negation appear within the pre or postcondition. In the example above, the existential quantifier ($\exists y$) and the modal ($\mathcal{O}$) appear within the postcondition.

Abstract Syntax Tree: The AST for (1) is shown in Figure 1. The main nodes of interest are the internal nodes labeled $\boxed{\lambda}$. An internal node with $n+1$ children corresponds to an *n-ary operator*. The first child of the internal node is the operator. Operators are labeled with a part-of-speech tag, for example, "D" for determiner, "M" for modal, and "O" for other. The remaining n children are its arguments. We use the term *nuclear scope* to refer to the last (n^{th}) argument of the operator, and the term *restrictor* to refer to any other argument. We borrow these terms from the literature on quantifier scope for determiners (Heim and Kratzer, 1998, Chapter 7).

For example, the phrase "general safety test" is in the restrictor of the operator A, and the variable $\boxed{y}$ is in its nuclear scope. The modal shall is a unary operator, and doesn't have a restrictor. Non-unary operators bind the variable displayed on the internal node. The variable $\boxed{y}$ is bound by the operator A.

Implicit operators are inserted when there is no overt word or phrase. In Figure 1, the implicit operators are underlined. The generic noun phrase "biological products" is associated with the implicit determiner all. Similarly, we use the implicit operator Post to mark the position of the postcondition.

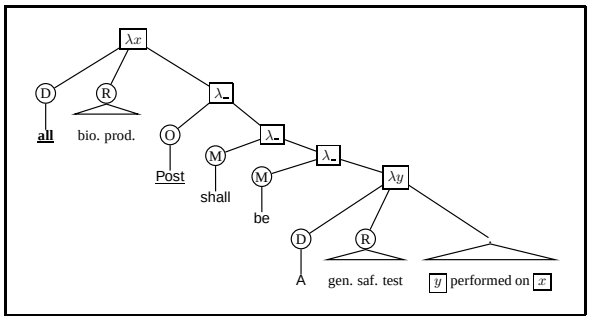


Figure 1: Example of an abstract syntax tree (AST).

We conclude this section with some notation for describing ASTs. Given an AST for a sentences, we say that an operator o_i *scopes over* o_j, denoted $o_i \gg o_j$, if o_j appears in the nuclear scope of o_i. For example, in Figure 1, we have all $\gg$ Post, all $\gg$ shall, all $\gg$ A, Post $\gg$ A, and shall $\gg$ A. In addition, we say that *the restrictor of* o_i *scopes over* o_j, denoted $R(o_i) \gg o_j$, if o_j appears in the restrictor of o_i. Such configurations occur with PP-modification of NPs, and we discuss examples in later sections.

3 Computing ASTs – Overview

In this section, we give an overview of our approach to computing ASTs. We will assume as given a Processed Parse Tree (PPT) of a sentence, with the operators and their restrictors identified. An example is discussed in Section 3.1. Given such a PPT, the AST is computed in two steps: (1) finding the preterminal at which an operator takes scope, and (2) ordering the operators associated with a preterminal. We describe the second step in Section 3.2, and then briefly outline the first step in Section 3.3. The steps are described in reverse order, because in most cases, the operators associated with a preterminal are determined directly by syntactic attachment.

3.1 Processed Parse Trees

We compute ASTs from processed parse trees (PPTs) of sentences. Figure 2 gives the PPT corresponding to the AST in Figure 1.

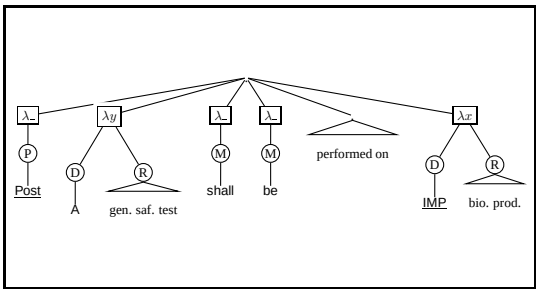


Figure 2: Processed parse tree (PPT) for (1).

A PPT provides the set of operators in a sentence, associated with their restrictors. For example, the determiner "a" has the restrictor *general safety test*. The phrase *biological products* has no explicit determiner associated with it, and the corresponding operator in the PPT is labeled "IMP" for *implicit*. In addition, the postcondition marker "Post" is also identified. Except for the postcon-

dition marker, annotator-specified implicit operators are not given in the PPT.

There are two main types of nodes in the PPT – *operators* and *preterminals*. The nodes labeled with the symbol λ, e.g., $\boxed{\lambda\_}$ and $\boxed{\lambda x}$, correspond to operators. The root of the PPT and the restrictors of the operators, are the preterminals. Based on this example, it may seem that a sentence just has a list of operators. While this is true of example (1), embedded operators arise, for example, in the context of PP-modification of NPs and relative clauses. We will discuss an example in Section 3.3.

In this work, the PPTs are obtained by removing all scope decisions from the AST. To a first approximation, we start by removing all operators from the AST, and then, replace the corresponding variables by the operators. Implicit unary operators (such as the postcondition marker) are placed at the start of the preterminal.

It is worthwhile to consider whether it is reasonable to assume PPTs as given. We believe that this assumption is (slightly) stronger than assuming perfect parse trees. Although the PPT leaves certain chunks of the sentence unprocessed, in most cases, the unprocessed chunks correspond to base NPs. The main additional piece of information is the existence of a postcondition marker for each main clause of a sentence. We believe that computation of PPTs is better seen as a problem of syntax rather than scope, and we set it aside to future work. Our focus here is on converting a PPT to an AST.

3.2 Ordering Operators

The problem of learning to order a set of items is not new. Cohen et al. (1998) give a learning theoretic perspective, and Liu (2009) surveys information retrieval applications. The approach that we use can be seen as a probabilistic version of the boosting approach developed by Cohen et al. (1998). We explain the step of ordering operators, by revisiting the example of the general safety test, from Section 2.

Given the PPT in Figure 2, we compute the AST in Figure 1 by *ordering* or *ranking* the operators. For example, we need to determine that the implicit determiner associated with *biological products* is universal, and hence, we have $\underline{\mathsf{IMP}} \gg \underline{\mathsf{Post}}$. However, the determiner "A" associated with *general safety test* is existential, and hence, we have $\underline{\mathsf{Post}} \gg \mathsf{A}$.

We now develop some notation to describe the scopal ordering of operators. A PPT τ is viewed as a set of preterminal nodes, and we will write – (a) $\mathfrak{p} \in \tau$ to denote that $\mathfrak{p}$ occurs in τ, and (b) $|\tau|$ to denote the number of preterminals in τ. A preterminal $\mathfrak{p}$ is viewed as an ordered set of operators $\mathfrak{p} = (\mathsf{o}^1, ..., \mathsf{o}^{|\mathfrak{p}|})$. For example, in Figure 2, the root preterminal $\mathfrak{p}$ has $|\mathfrak{p}| = 5$, and the operators $\mathsf{o}^1 = \underline{\mathsf{Post}}$, $\mathsf{o}^2 = \mathsf{A}$, $\mathsf{o}^3 = \mathsf{shall}$, and so on.

An AST α contains a ranking of operators associated with each preterminal, denoted $\mathfrak{r}_\alpha(\mathfrak{p})$. The ranks of operators are denoted by subscripts. Let $\mathfrak{p} = (\mathsf{o}^1, ..., \mathsf{o}^5)$ be the root preterminal of the PPT in Figure 2. The ranking associated with the AST in Figure 1 is given by $\mathfrak{r}_\alpha(\mathfrak{p}) = (\mathsf{o}^1_2, \mathsf{o}^2_5, \mathsf{o}^3_3, \mathsf{o}^4_4, \mathsf{o}^5_1)$. For example, $\mathsf{o}^2_5 = \mathsf{A}$ denotes that the determiner "A" appears second in the surface order (Figure 2) and fifth or lowest in the scope order (Figure 1). Similarly, $\mathsf{o}^5_1 = \underline{\mathsf{IMP}}$ denotes that the implicit determiner appears fifth or last in the surface order (Figure 2) and first or highest in the scope order (Figure 1). Note that the subscript suffices to identify the position of an operator in the AST.

Model: We now describe the learning model for ordering operators. Given a PPT τ, let $\mathcal{A}(\tau)$ be the set of all possible ASTs. Our goal is to find the AST which has the highest probability given the PPT:

$$\alpha^* = \arg \max_{\alpha \in \mathcal{A}(\tau)} P(\alpha|\tau)$$

The conditional probability of an AST is defined as:

$$P(\alpha|\tau) = \prod_{\mathfrak{p} \in \tau} P(\mathfrak{r}_\alpha(\mathfrak{p})|\tau)$$

$$P(\mathfrak{r}_\alpha(\mathfrak{p})|\tau) = \prod_{i=1}^{|\mathfrak{p}|-1} \prod_{j=i+1}^{|\mathfrak{p}|} P(\mathsf{o_i} \gg \mathsf{o_j}|\tau)$$

In other words, $P(\alpha|\tau)$ is modeled as the product of the probabilities of the ranking of each preterminal, which is in turn expressed as the product of the probabilities of the pairwise ordering decisions. The model falls under the class of pairwise ranking approaches (Liu, 2009). We will consider the problem of estimating the probabilities in Section 5, and the problem of searching for the best AST in Section 6.

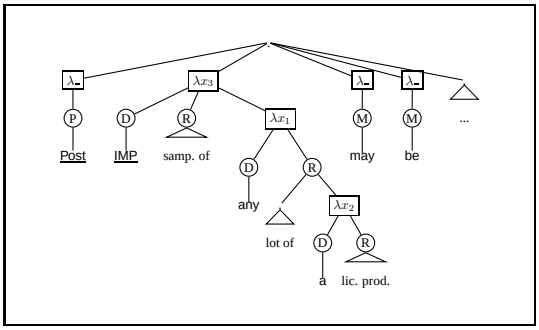


Figure 3: PPT for (2)

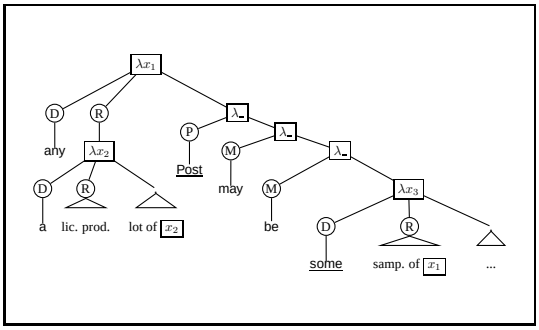


Figure 4: AST for (2)

3.3 Finding the Scope Preterminal

In the example that we discussed in the previous section, there were no embedded operators, i.e., an operator or its variable located in the restrictor of another. An embedded operator can either – (a) take scope within the restrictor of the embedding operator, or (b) outscope the embedding operator. To account for the second case, we need to determine whether it is appropriate to lift an embedded operator to a higher preterminal than the one to which it is associated syntactically.

We discuss an example of *inverse linking* (Larson, 1985) to illustrate the problem. Consider the following sentence:

(2) Samples of any lot of a licensed product, except for radioactive biological products, together with the protocols showing results of applicable tests, may at any time be required to be sent to the Director, Center for Biologics Evaluation and Research.

The PPT and AST for (2) are shown in Figures 3 and 4 respectively. Consider the noun phrase "IMP samples of any lot of a licensed product" in the PPT. The implicit determiner IMP in the PPT is interpreted as the existential determiner some in the AST. The three operators are related as follows in the AST: any $\gg$ some and R(any) $\gg$ a, i.e., any outscopes the implicit determiner, and a appears in the restrictor of any. Observe that the variables x_1 and x_2, which are associated with any and a, appear in the restrictors of some and any respectively. As a result, in the PPT, in Figure 3, any and a appear in the restrictor of IMP and any. The PPT provides a standard parse of PP-modification of NPs.

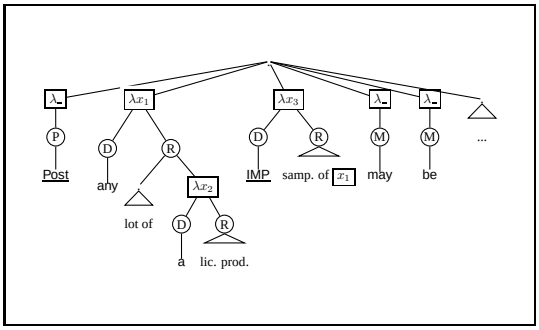


Figure 5: Second PPT for (2), obtained from the PPT in Figure 3, by raising any to the root preterminal.

The important feature of this example is that the determiner "any" is syntactically embedded in the restrictor of IMP in the PPT (Figure 4), but it outscopes the implicit determiner in the AST (Figure 3). As a result, the PPT in Figure 3 cannot be converted to the AST in Figure 4 simply by ranking sibling operators (as we did in the previous section).

To handle such cases, we convert the PPT in Figure 3 to a second PPT (shown in Figure 5). The only allowed operation during this conversion is to raise an embedded operator to a higher preterminal. The PPT in Figure 5 is obtained by raising any to the root preterminal, making it a sibling of the implicit determiner IMP in the PPT in Figure 5. This second PPT can be converted to the AST by reordering sibling operators. The learning model used for this step is similar to the one used to order operators, and in the interests of space, we omit the details.

4 Brief Overview of the Corpus

We have annotated 195 sentences from the FDA CFR Section 610 with ASTs. The operators are divided into the following types – determiners (e.g.,

every, a, at least), modal auxiliaries (e.g., must, be), VP modifiers (e.g., if, for, after), negation and coordinating conjunctions (e.g., and, but, or). The majority of the corpus was annotated by a single annotator. However, to estimate inter-annotator agreement, a set of 32 sentences was annotated by a second annotator. In this section, we restrict ourselves to presenting statistics that highlight part of the guidelines and motivate the features that we use to order operators. An example-based justification of guidelines, and a discussion of inter-annotator agreement can be found in (Dinesh, 2010).

De Re vs De Dicto: We narrow our focus to one part of the annotation, the *de re* vs *de dicto* distinction. Informally, operators with *de re* scope occur in the precondition of the logical translation of a sentence, while those with *de dicto* scope occur in the postcondition. This distinction is of key importance in the application of conformance checking, as it helps determine the facts that need to be provided by an organization (*de re*), and the actions that an organization is required to take (*de dicto*).

For simplicity, we further restrict attention to operators that are siblings of the postcondition in the AST, and ignore the operators embedded in prepositional phrases and clauses, for example. A (main clause) operator o is said to have *de re* scope iff it outscopes the postcondition marker ($o \gg \underline{Post}$). Otherwise, the operator is said to have *de dicto* scope ($\underline{Post} \gg o$). In the example of the general safety test from Section 2, the implicit determiner associated with "biological products" has *de re* scope, while all other operators in the sentence have *de dicto* scope.

Operator Type	Number of Instances	De Re Scope Percentage
Determiner	277	59.9%
Modal Aux	268	0%
VP Modifier	132	68.2%
CC	36	22.2%
Neg	33	0%
Other	74	17.6%

Table 1: De Re scope distribution. An operator has *de re* scope iff it outscopes the postcondition marker.

Table 1 shows the percentage of each type of operator that has *de re* scope. Modal auxiliaries and negation are umambigous to this distinction, and always have *de dicto* scope. Note that a type of operator with 50% occuring *de re* is ambiguous, while 0% or 100% are unambiguous. Thus, from Table 1, we can conclude that determiners, and VP modifiers are the most ambiguous types. And, more features are needed to disambiguate them.

Determiner Type	Number of Instances	De Re Scope Percentage
Universal	74	100%
Existential	12	0%
Ambiguous	50	28%
Deictic	127	53.5%
Other	14	35.7%

Table 2: De Re scope distribution for determiners.

Determiners: We divide the determiners into the following subtypes: universal/generic (e.g., every, all), existential (some), ambiguous (e.g., a, an), deictic (e.g., the, those), and other (e.g., at least, at most). The guidelines for annotation were as follows – (a) universal determiners have *de re* scope, (b) existential determiners have *de dicto* scope, and (c) for other determiners, the annotator needs to decide whether a particular use is interpreted existentially or universally. Table 2 shows *de re* scope distribution for each of these subtypes. As expected, universal and existential determiners are unambiguous, while ambiguous and deictic determiners show more variety. For example, the deictic determiner the can refer to a specific entity ("the FDA") or have a universal interpretation ("the products").

Thus, to disambiguate between *de re* and *de dicto* interpretations for determiners, we need two types of features – (1) Features to predict whether ambiguous and deictic determiners are universal or not, and (2) Features to determine the type of implicit determiners. In Table 2, we assume that the type of implicit determiners are given. This assumption is unrealistic. Rather, we need to predict the type of such determiners, during the computation of the AST.

VP Modifier Type	Number of Instances	De Re Scope Percentage
Temporal and Conditional	73	100%
Purpose	8	0%
References to Laws	33	0.9%
Other	29	65.5%

Table 3: De Re scope distribution for VP modifiers.

VP Modifiers: We divide the VP modifiers into the following subtypes: temporal and conditional (e.g., after, if), purpose (for), references to laws (which are a special type of modifier in the legal domain, e.g., "as specified in paragraph (c)"), and other (e.g., regardless, notwithstanding). Table 3 shows the percentage of each subtype of modifier that has *de re* scope. Following the guidelines for annotation, the temporal and conditional modifiers are always *de re*, the purpose modifiers and modifiers conveying references to laws are always *de dicto*.

5 Comparing the Scope of Operators

We now consider a subproblem in computing the AST – comparing the scope of pairs of operators. In Section 6, we will use the classifiers that perform comparisons, to compute the AST. All experiments in this section use the MAXENT implentation from the MALLET toolkit (McCallum, 2002). We begin by revisiting *de re-de dicto* distinction from Section 4. Then, we generalize to other comparisons.

De Re vs De Dicto: The (binary) classification problem is as follows. Our observations are triples $x = (\mathsf{o}, \mathsf{o}', \tau)$ are such that there is a preterminal $\mathfrak{p} \in \tau$, $\{\mathsf{o}, \mathsf{o}'\} \subseteq \mathfrak{p}$, and $\mathsf{o}' = \underline{\mathsf{Post}}$. In other words, we are considering operators (o) that are siblings of the postcondition marker (o'). An observation has the label 1 if $\mathsf{o} \gg \mathsf{o}'$ (*de re* scope), and a label of 0 otherwise (*de dicto* scope).

Features: We use the following (classes of) features for an observation $x = (\mathsf{o}, \mathsf{o}', \tau)$:

- TYPE - The type and subtype of the operator. We use the subtypes from Section 4 only for explicit operators.
- PRE-VERB - Tracks whether o and o' appear before or after the main verb of the sentence.
- PRE-VERB + PERF - Conjunction of the previous feature with whether the main verb is *perform*. The verb *perform* is frequent in the CFR, and its subject is typically given *de dicto* scope, as it is the main predicate of the sentence.
- POS - The part-of-speech of the head word. For example, for the noun phrase *biological products*, the head word is *products*, and the POS is NNS (plural common noun). And, this POS tag may indicate a generic/universal interpretation.

	Count	MAJORITY	TYPE	ALL
All	823	66.2%	84.1%	89.2%
No MD	522	53.2%	74.9%	83.7%
DT	277	59.9%	62.9%	81.2%
Imp. DT	100	69%	-	76%

Table 4: De Re vs De Dicto classification. Average accuracies over 10-fold cross-validation. The rows describe the subset of observations considered, and the columns describe the subset of features used.

Experiments: We evaluate the features by performing 10-fold cross-validation. The results are summarized in Table 4. The rows describe the subset of observations used. "All" includes all observations, "No MD" excludes the modal auxiliaries, "DT" includes only the determiners, and "Imp. DT" includes only implicit determiners. The columns describe the features used. MAJORITY is the majority baseline, i.e., the accuracy obtained by predicting the most frequent class or the majority class. The majority class is *de dicto* when all operators are considered (the first row), and *de re* in all other rows. The TYPE column gives the accuracy when only the type and subtypes are used as features. This column does not apply to implicit determiners, as the subtype information is unavailable. And, finally, the ALL column gives the accuracy when all features are used.

From Table 4, we can conclude that the TYPE feature is useful in making the *de re-de dicto* distinction, and further gains are obtained by using ALL features. The most dramatic improvement is for determiners, and indeed, our features were designed for this case. However, the performance gains are not very high for implicit determiners, and further investigation is needed.

Next, we apply the features to more general operator comparisons. The first row of Table 5 considers observations $x = (\mathsf{o}, \mathsf{o}', \tau)$, where o and o' are siblings, and predicts whether $\mathsf{o} \gg \mathsf{o}'$. The second row considers observations where o' is embedded syntactically within o, and predicts whether $\mathrm{R}(\mathsf{o}) \gg \mathsf{o}'$. In other words, the problem is to determine whether a syntactically embedded operator remains scopally embedded, or whether it has inverse scope (see Section 3.3).

	Count	MAJORITY	TYPE	ALL
Siblings	2793	76.1%	83.3%	87.5%
Embedded	5081	95%	95.3%	96.4%

Table 5: Ordering siblings and embedded operators. Average accuracies over 10-fold cross-validation. The columns describe the subset of features used.

6 From Operator Comparisons to ASTs

We now consider the problem of computing the AST given the classifiers for comparing operators. Section 6.1 describes the algorithms used. In Section 6.2, we develop metrics to evaluate the computation of ASTs. We conclude, in Section 6.3, by evaluating different algorithms using the metrics.

6.1 Algorithms

We begin by discussing the intractability of the problem of ranking or ordering operators. Then, we sketch the search heuristics used.

Intractability: The decision version of the ranking problem is NP-complete. A similar result is established by Cohen et al. (1998) in the context of a boosting approach to ranking.

Theorem 1. *The following problem is NP-complete:*
Input: *A PPT τ, a preterminal $\mathfrak{p} \in \tau$, probabilities $P(\mathfrak{o}^i \gg \mathfrak{o}^j|\tau)$, and $c \in [0,1]$*
Output: *Yes, if there is an ordering $\mathfrak{r}$ such that $P(\mathfrak{r}(\mathfrak{p})|\tau) \geq c$*

The proof is by reduction from ACYCLIC SUBGRAPH (Karp, 1972) – finding a subgraph which is acyclic and has at least k edges.

Heuristics: To order operators, we use a beam search procedure. Each search state consists preterminal, in which the first i ranks have been assigned to operators. We then search over next states by assigning the rank $i+1$ to one of the remaining operators. We used a beam size of 10^4 in our experiments. In most cases, the number of operators per preterminal is less than 7. As a result, the total number of possible orderings is typically less than 7!, and a beam size of 10^4 is sufficient to compute an exact ordering. In other words, due to the size restrictions, in most cases, beam search is equivalent to exact (exhaustive) search.

To handle embedded operators, we use a simple greedy heuristic. We enumerate the operators in the initial PPT, corresponding to an in-order traversal. For each operator, we attach it to the most likely ancestor, given the attachment decisions for the previous operators. This heuristic is optimal for the case where the depth of embedding is at most 1, which is the common case.

6.2 Metrics

In this section, we describe metrics used to evaluate the computation of ASTs. Let τ be the initial PPT, α the correct AST, and α^* the computed AST. We define accuracy at various levels.

The simplest metric is to define accuracy at *the level of ASTs*, i.e., by computing the fraction of cases for which $\alpha = \alpha^*$. However, this metric is harsh, in the sense that it does not give algorithms partial credit for getting a portion of the AST correct.

The next possible metric is to define accuracy at *the level of preterminals*. Let $\mathfrak{p}$ be a preterminal. Note that τ, α and α^* share the same set of preterminals, but may associate different operators with them. We say that $\mathfrak{p}$ is correct in α^*, if it is associated with the same set of operators as in α, and for all $\{\mathfrak{o}, \mathfrak{o}'\} \subseteq \mathfrak{p}$, we have $\mathfrak{o} \gg \mathfrak{o}'$ w.r.t. α^* iff $\mathfrak{o} \gg \mathfrak{o}'$ w.r.t. α. In other words, the preterminals are identical, both in terms of the set of operators and the ordering between pairs of operators. While preterminal-level accuracy gives partial credit, it is still a little harsh, in the sense that an algorithm which makes one ordering mistake at a preterminal is penalized the same as an algorithm which makes multiple mistakes.

Finally, we consider metrics to define accuracy at *the level of pairs of operators*. Let $\mathfrak{p}$ be a preterminal. The set $\mathrm{Pairs}(\mathfrak{p}, \alpha)$ consists of pairs of operators $(\mathfrak{o}, \mathfrak{o}')$ such that $\mathfrak{o}$ and $\mathfrak{o}'$ are both associated with $\mathfrak{p}$ in α, and $\mathfrak{o} = \mathfrak{o}'$ or $\mathfrak{o} \gg \mathfrak{o}'$. The set $\mathrm{Pairs}(\mathfrak{p}, \alpha^*)$ is defined similarly using α^* instead of α. Given the sets $\mathrm{Pairs}(\mathfrak{p}, \alpha)$ and $\mathrm{Pairs}(\mathfrak{p}, \alpha^*)$, precision, recall, and f-score are defined in the usual way. We leave the details to the reader.

6.3 Results

We evaluate the following algorithms:

1. No Embedding – The AST is computed purely by reordering operators within a preterminal in the PPT.

(a) SURFACE – No reordering is performed, i.e., the order of operators in the AST respects the surface order

(b) TYPE – Using only type and subtype information for the operators

(c) ALL – Using all the features described in Section 5

2. ALL+ – The initial PPT is transformed into a second PPT before reordering (as described in Section 3.3). All features are used.

	Prec.	Rec.	F	$\mathfrak{p}$	α
SURF.	86.9%	82.7%	84.6%	81%	4.2%
TYPE	90.4%	86%	88.1%	83.6%	24.7%
ALL	**92**%	87.6%	89.8%	85.1%	33.5%
ALL+	91.9%	**89.4**%	**90.6**%	**85.9**%	**36.2**%

Table 6: Performance of the algorithms in computing the ASTs. Averaged over 10-fold cross-validation. 195 ASTs in total, an average of 8.6 preterminals per AST, and 1.8 operators per preterminal.

Table 6 summarizes the performance of the algorithms, under the various metrics. The accuracies are averaged over 10-fold cross-validation. A total of 195 ASTs are used. The average number of preterminals per AST is 8.6, with an average of 1.8 operators per preterminal. The best number under each metric is shown in bold-face. By adding features, we improve the precision from 86.9% to 90.4% to 92% in moving from SURFACE to TYPE to ALL. By handling embedded operators, we improve the recall from 87.6% to 89.4% in moving from ALL to ALL+. As we saw in Section 5, in 95% of the cases, the embedded operators respects syntactic scope, and as a result, we obtain only modest gains from handling embedded operators.

The reader may feel that the F-score of 90.6% is quite high given the size of our training data. This score is inflated by inclusion of reflexive pairs, of the form (o, o). Such pairs are included for the following (technical) reasons. The algorithm that handles embedded operators (ALL+) usually raises them from a single operator node (as in Figure 3) to a multi-operator node (as in Figure 5). If it makes an incorrect decision to raise an operator it takes a precision hit, at the multi-operator node (because it has some false positives). By contrast, an algorithm loses precision for failing to correctly raise, only when we encounter the single operator node.

For these reasons, it is better to consider the relative improvement in F-score over the baseline. The relative improvement of ALL+ over SURFACE in terms of F-score is 36.6%. We believe that the preterminal-level accuracy is more indicative in an absolute sense. Furthermore, when we restrict attention to those preterminals with two or more operators in the PPT, the accuracy of ALL+ is 69.4%.

7 Related Work

ASTs can be seen as a middle ground between two lines of research in translating sentences to logic.

At one end of the spectrum, we have methods that achieve good accuracy on restricted texts. The two main corpora that have been considered are the GEOQUERY corpus (Thompson et al., 1997) and the ATIS-3 corpus (Dahl et al., 1994). The GEOQUERY corpus consists of queries to a geographical database. The queries were collected from students participating in a study and the average sentence length is 8 words. The ATIS corpus is collected from subjects' interaction with a database of flight information, using spoken natural language. The utterances have be transcribed, and the average sentence length is 10 words (Berant et al., 2007). Algorithms, which achieve good accuracy, have been developed to compute the logical translation for these queries (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Zettlemoyer and Collins, 2009). The annotated sentences in the FDA CFR Section 610.40 are longer (about 30 words on average), and contain modalities which are not present in these corpora.

At the other end of the spectrum, Bos et al. (Bos et al., 2004) have developed a broad-coverage parser to translate sentences to a logic based on discourse representation theory. Here, there is no direct method to evaluate the correctness of the translation. However, indirect evaluations are possible, for example, by studying improvement in textual entailment tasks.

To summarize, there are techniques that either produce an accurate translation for sentences in a limited domain, or produce some translation for sentences in a broader range of texts. ASTs offer a middle ground in two ways. First, we focus on regula-

tory texts which are less restricted than the database queries in the GEOQUERY and ATIS corpora, but do not exhibit anaphoric phenomenon found in genres, such as, newspaper text. In (Dinesh et al., 2007), we discuss lexical statistics that show significant differences in the distribution of anaphoric items in the CFR and Wall Street Journal (WSJ) corpora. For example, the frequency of pronouns and anaphoric discourse connectives is significantly lower in the CFR than in the WSJ. Instead, the CFR has an idiosyncratic mechanism for referring to sentences, using phrases such as "except as specified in paragraph (c) and (d)". A question of interest is whether the GEOQUERY and ATIS corpora show similar peculiarities in terms of anaphora. The second difference between our approach and others is that we do not attempt to translate all the way to logic. The level of logical form lets us obtain a direct evaluation, while leaving open the design of parts of the logic.

8 Conclusions

We described experiments on a modest-sized corpus of regulatory sentences, annotated with a novel variant of logical form, called *abstract syntax trees* (ASTs). An example from the corpus was presented in Section 2 and some statistics, describing the corpus, were discussed in Section 4. In Sections 3, 5, and 6, we developed and tested algorithms to convert a processed parse tree (PPT) to an AST. The main step in this conversion was to rank or order the operators at a preterminal. We presented a probabilistic model for ranking, investigated the design of features, and developed search heuristics. The best algorithm, which uses all features and handles embedded operators, achieves an F-score of 90.6%.

An important direction for further inquiry is in the design of better features. Various types of features have been proposed for the scopal ordering of determiners. Examples include syntactic features (Ioup, 1975; Reinhart, 1983), such as position and voice, semantic features (Grimshaw, 1990; Jackendoff, 1972), such as thematic roles. More recently, Srinivasan and Yates (2009) showed how pragmatic information, for example "there are more people than cities", can be leveraged for scope disambiguation. We experimented with lexico-syntactic features in this work, and leave an investigation of semantic and pragmatic features to future work.

Acknowledgements

We thank Claire Cardie, Steve Kimbrough, Annie Louis, Fernando Pereira, Emily Pitler, Oleg Sokolsky, and the anonymous reviewers for helpful comments on earlier versions of this paper.

References

A. Aho, R. Sethi, and J. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Addison-Wessley.

R. J. Anderson. 1996. A security policy model for clincial information systems. In *Proceedings of the IEEE Symposium on Security and Privacy*.

A. Barth, A. Dutta, J. C. Mitchell, and H. Nissenbaum. 2006. Privacy and contextual integrity: Framework and applications. In *Proceedings IEEE Symposium on Security and Privacy*.

J. Berant, Y. Gross, M. Mussel, B. Sandbank, E. Ruppin, and S. Edelman. 2007. Boosting unsupervised grammar induction by splitting complex sentences on function words. In *Proceedings of the Boston University Conference on Language Development*.

J. Bos, S. Clark, M. Steedman, J. R. Curran, and J. Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING*.

W. W. Cohen, R. E. Schapire, and Y. Singer. 1998. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270.

D. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. 1994. Expanding the scope of the ATIS task: the ATIS-3 corpus. In *Proceedings of the ARPA HLT Workshop*.

N. Dinesh, A. Joshi, I. Lee, and O. Sokolsky. 2007. Logic-based regulatory conformance checking. In *Proceedings of the 14th Monterey Workshop*.

N. Dinesh, A. Joshi, I. Lee, and O. Sokolsky. 2011. Permission to speak: A logic for access control and conformance. *Journal of Logic and Algebraic Programming*, 80(1):50–74.

N. Dinesh. 2010. *Regulatory Conformance Checking: Logic and Logical Form*. Ph.D. thesis, University of Pennsylvania.

G. Governatori, Z. Milosevic, and S. Sadiq. 2006. Compliance checking between business processes and business contracts. In *10th International Enterprise Distributed Object Computing Conference (EDOC)*.

J. Grimshaw. 1990. *Argument Structure*. MIT Press.

B. Grosof, Y. Labrou, and H. Y. Chan. 1999. A declarative approache to business rules in contracts: Courteous logic programs in xml. In *ACM Conference on Electronic Commerce*.

Irene Heim and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell.

G. Ioup. 1975. Some universals for quantifier scope. *Syntax and Semantics*, 4:37–58.

R. Jackendoff. 1972. *Semantic Interpretation in Generative Grammar*. MIT Press.

A. J. I. Jones and M. J. Sergot. 1992. Formal specification of security requirements using the theory of normative positions. In *European Symposium on Reasearch in Computer Security (ESORICS)*.

R. M. Karp. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

H. S. Kurtzman and M. C. MacDonald. 1993. Resolution of quantifier scope ambiguities. *Cognition*, 48:243–279.

R. K. Larson. 1985. Quantifying to np. Manuscript, MIT.

T. Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3).

D. Makinson and L. van der Torre. 2000. Input/output logics. *Journal of Philosophical Logic*, 29:383–408.

R. May. 1985. *Logical Form: Its structure and derivation*. MIT Press.

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

T. Reinhart. 1983. *Anaphora and Semantic Interpretation*. Croom Helm.

M.J. Sergot, F.Sadri, R.A. Kowalski, F.Kriwaczek, P.Hammond, and H.T. Cory. 1986. The british nationality act as a logic program. *Communications of the ACM*, 29(5):370–86.

P. Srinivasan and A. Yates. 2009. Quantifier scope disambiguation using extracted pragmatic knowledge: Preliminary results. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

C. A. Thompson, R. J. Mooney, and L. R. Tang. 1997. Learning to parse natural language database queries into logical form. In *Proceedings of the Workshop on Automata Induction, Grammatical Inference and Language Acquisition*.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

L. S. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Computation of Infix Probabilities for Probabilistic Context-Free Grammars

Mark-Jan Nederhof
School of Computer Science
University of St Andrews
United Kingdom
markjan.nederhof@gmail.com

Giorgio Satta
Dept. of Information Engineering
University of Padua
Italy
satta@dei.unipd.it

Abstract

The notion of infix probability has been introduced in the literature as a generalization of the notion of prefix (or initial substring) probability, motivated by applications in speech recognition and word error correction. For the case where a probabilistic context-free grammar is used as language model, methods for the computation of infix probabilities have been presented in the literature, based on various simplifying assumptions. Here we present a solution that applies to the problem in its full generality.

1 Introduction

Probabilistic context-free grammars (PCFGs for short) are a statistical model widely used in natural language processing. Several computational problems related to PCFGs have been investigated in the literature, motivated by applications in modeling of natural language syntax. One such problem is the computation of **prefix probabilities** for PCFGs, where we are given as input a PCFG $\mathcal{G}$ and a string w, and we are asked to compute the probability that a sentence generated by $\mathcal{G}$ starts with w, that is, has w as a prefix. This quantity is defined as the possibly infinite sum of the probabilities of all strings of the form wx, for any string x over the alphabet of $\mathcal{G}$.

The problem of computation of prefix probabilities for PCFGs was first formulated by Persoon and Fu (1975). Efficient algorithms for its solution have been proposed by Jelinek and Lafferty (1991) and Stolcke (1995). Prefix probabilities can be used to compute probability distributions for the next word or part-of-speech, when a prefix of the input has already been processed, as discussed by Jelinek and Lafferty (1991). Such distributions are useful for speech recognition, where the result of the acoustic processor is represented as a lattice, and local choices must be made for a next transition. In addition, distributions for the next word are also useful for applications of word error correction, when one is processing 'noisy' text and the parser recognizes an error that must be recovered by operations of insertion, replacement or deletion.

Motivated by the above applications, the problem of the computation of **infix probabilities** for PCFGs has been introduced in the literature as a generalization of the prefix probability problem. We are now given a PCFG $\mathcal{G}$ and a string w, and we are asked to compute the probability that a sentence generated by $\mathcal{G}$ has w as an infix. This probability is defined as the possibly infinite sum of the probabilities of all strings of the form xwy, for any pair of strings x and y over the alphabet of $\mathcal{G}$. Besides applications in computation of the probability distribution for the next word token and in word error correction, infix probabilities can also be exploited in speech understanding systems to score partial hypotheses in algorithms based on beam search, as discussed by Corazza et al. (1991).

Corazza et al. (1991) have pointed out that the computation of infix probabilities is more difficult than the computation of prefix probabilities, due to the added ambiguity that several occurrences of the given infix can be found in a single string generated by the PCFG. The authors developed solutions for the case where some distribution can be defined on

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1213–1221,
Edinburgh, Scotland, UK, July 27–31, 2011.

the distance of the infix from the sentence boundaries, which is a simplifying assumption. The problem is also considered by Fred (2000), which provides algorithms for the case where the language model is a probabilistic regular grammar. However, the algorithm in (Fred, 2000) does not apply to cases with multiple occurrences of the given infix within a string in the language, which is what was pointed out to be the problematic case.

In this paper we adopt a novel approach to the problem of computation of infix probabilities, by removing the ambiguity that would be caused by multiple occurrences of the given infix. Although our result is obtained by a combination of well-known techniques from the literature on PCFG parsing and pattern matching, as far as we know this is the first algorithm for the computation of infix probabilities that works for general PCFG models without any restrictive assumption.

The remainder of this paper is structured as follows. In Section 2 we explain how the sum of the probabilities of all trees generated by a PCFG can be computed as the least fixed-point solution of a non-linear system of equations. In Section 3 we recall the construction of a new PCFG out of a given PCFG and a given finite automaton, such that the language generated by the new grammar is the intersection of the languages generated by the given PCFG and the automaton, and the probabilities of the generated strings are preserved. In Section 4 we show how one can efficiently construct an unambiguous finite automaton that accepts all strings with a given infix. The material from these three sections is combined into a new algorithm in Section 5, which allows computation of the infix probability for PCFGs. This is the main result of this paper. Several extensions of the basic technique are discussed in Section 6. Section 7 discusses implementation and some experiments.

2 Sum of probabilities of all derivations

Assume a probabilistic context-free grammar $\mathcal{G}$, represented by a 5-tuple (Σ, N, S, R, p), where Σ and N are two finite disjoint sets of **terminals** and **nonterminals**, respectively, $S \in N$ is the **start symbol**, R is a finite set of **rules**, each of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$, and p is a function from rules in R to real numbers in the interval $[0, 1]$.

The concept of **left-most derivation** in one step is represented by the notation $\alpha \stackrel{\pi}{\Rightarrow}_{\mathcal{G}} \beta$, which means that the left-most occurrence of any nonterminal in $\alpha \in (\Sigma \cup N)^*$ is rewritten by means of some rule $\pi \in R$. If the rewritten nonterminal is A, then π must be of the form $(A \to \gamma)$ and β is the result of replacing the occurrence of A in α by γ. A left-most derivation with any number of steps, using a sequence d of rules, is denoted as $\alpha \stackrel{d}{\Rightarrow}_{\mathcal{G}} \beta$. We omit the subscript $\mathcal{G}$ when the PCFG is understood. We also write $\alpha \stackrel{*}{\Rightarrow} \beta$ when the involved sequence of rules is of no relevance. Henceforth, all derivations we discuss are implicitly left-most.

A **complete** derivation is either the empty sequence of rules, or a sequence $d = \pi_1 \cdots \pi_m$, $m \geq 1$, of rules such that $A \stackrel{d}{\Rightarrow} w$ for some $A \in N$ and $w \in \Sigma^*$. In the latter case, we say the complete derivation starts with A, and in the former case, with d an empty sequence of rules, we assume the complete derivation starts and ends with a single terminal, which is left unspecified. It is well-known that there exists a bijective correspondence between left-most complete derivations starting with nonterminal A and parse trees derived by the grammar with root A and a yield composed of terminal symbols only.

The **depth** of a complete derivation d is the length of the longest path from the root to a leaf in the parse tree associated with d. The length of a path is defined as the number of nodes it visits. Thus if $d = \pi$ for some rule $\pi = (A \to w)$ with $w \in \Sigma^*$, then the depth of d is 2.

The probability $p(d)$ of a complete derivation $d = \pi_1 \cdots \pi_m$, $m \geq 1$, is:

$$p(d) \quad = \quad \prod_{i=1}^{m} p(\pi_i).$$

We also assume that $p(d) = 1$ when d is an empty sequence of rules. The probability $p(w)$ of a string w is the sum of all complete derivations deriving that string from the start symbol:

$$p(w) \quad = \quad \sum_{d:\ S \stackrel{d}{\Rightarrow} w} p(d).$$

With this notation, **consistency** of a PCFG is de-

fined as the condition:

$$\sum_{d,w:\ S \stackrel{d}{\Rightarrow} w} p(d) \quad = \quad 1.$$

In other words, a PCFG is consistent if the sum of probabilities of all complete derivations starting with S is 1. An equivalent definition of consistency considers the sum of probabilities of all strings:

$$\sum_{w} p(w) \quad = \quad 1.$$

See (Booth and Thompson, 1973) for further discussion.

In practice, PCFGs are often required to satisfy the additional condition:

$$\sum_{\pi=(A\rightarrow\alpha)} p(\pi) \quad = \quad 1,$$

for each $A \in N$. This condition is called **properness**. PCFGs that naturally arise by parameter estimation from corpora are generally consistent; see (Sánchez and Benedí, 1997; Chi and Geman, 1998). However, in what follows, neither properness nor consistency is guaranteed.

We define the **partition function** of $\mathcal{G}$ as the function Z that assigns to each $A \in N$ the value

$$Z(A) \quad = \quad \sum_{d,w} p(A \stackrel{d}{\Rightarrow} w). \tag{1}$$

Note that $Z(S) = 1$ means that $\mathcal{G}$ is consistent. More generally, in later sections we will need to compute the partition function for non-consistent PCFGs.

We can characterize the partition function of a PCFG as a solution of a specific system of equations. Following the approach in (Harris, 1963; Chi, 1999), we introduce generating functions associated with the nonterminals of the grammar. For $A \in N$ and $\alpha \in (N \cup \Sigma)^*$, we write $f(A, \alpha)$ to denote the number of occurrences of symbol A within string α. Let $N = \{A_1, A_2, \ldots, A_{|N|}\}$. For each $A_k \in N$, let m_k be the number of rules in R with left-hand side A_k, and assume some fixed order for these rules. For each i with $1 \leq i \leq m_k$, let $A_k \rightarrow \alpha_{k,i}$ be the i-th rule with left-hand side A_k.

For each k with $1 \leq k \leq |N|$, the **generating function** associated with A_k is defined as

$$g_{A_k}(z_1, z_2, \ldots, z_{|N|}) = \sum_{i=1}^{m_k} \Big(p(A_k \rightarrow \alpha_{k,i}) \cdot \prod_{j=1}^{|N|} z_j^{f(A_j, \alpha_{k,i})} \Big). \tag{2}$$

Furthermore, for each $i \geq 1$ we recursively define functions $g^{(i)}_{A_k}(z_1, z_2, \ldots, z_{|N|})$ by

$$g^{(1)}_{A_k}(z_1, z_2, \ldots, z_{|N|}) = g_{A_k}(z_1, z_2, \ldots, z_{|N|}), \tag{3}$$

and, for $i \geq 2$, by

$$\begin{aligned} &g^{(i)}_{A_k}(z_1, z_2, \ldots, z_{|N|}) = \\ &\quad g_{A_k}(\ g^{(i-1)}_{A_1}(z_1, z_2, \ldots, z_{|N|}), \\ &\qquad g^{(i-1)}_{A_2}(z_1, z_2, \ldots, z_{|N|}), \ldots, \\ &\qquad g^{(i-1)}_{A_{|N|}}(z_1, z_2, \ldots, z_{|N|})\ ). \end{aligned} \tag{4}$$

Using induction it is not difficult to show that, for each k and i as above, $g^{(i)}_{A_k}(0, 0, \ldots, 0)$ is the sum of the probabilities of all complete derivations from A_k having depth not exceeding i. This implies that, for $i = 0, 1, 2, \ldots$, the sequence of the $g^{(i)}_{A_k}(0, 0, \ldots, 0)$ monotonically converges to $Z(A_k)$.

For each k with $1 \leq k \leq |N|$ we can now write

$$\begin{aligned} &Z(A_k) = \\ &= \lim_{i\rightarrow\infty}\ g^{(i)}_{A_k}(0, \ldots, 0) \\ &= \lim_{i\rightarrow\infty}\ g_{A_k}(\ g^{(i-1)}_{A_1}(0, 0, \ldots, 0), \ldots, \\ &\qquad\qquad g^{(i-1)}_{A_{|N|}}(0, 0, \ldots, 0)\ ) \\ &= g_{A_k}(\ \lim_{i\rightarrow\infty}\ g^{(i-1)}_{A_1}(0, 0, \ldots, 0), \ldots, \\ &\qquad \lim_{i\rightarrow\infty}\ g^{(i-1)}_{A_{|N|}}(0, 0, \ldots, 0)\ ) \\ &= g_{A_k}(Z(A_1), \ldots, Z(A_{|N|})). \end{aligned}$$

The above shows that the values of the partition function provide a solution to the system of the following equations, for $1 \leq k \leq |N|$:

$$z_k \quad = \quad g_{A_k}(z_1, z_2, \ldots, z_{|N|}). \tag{5}$$

In the case of a general PCFG, the above equations are non-linear polynomials with positive (real) coefficients. We can represent the resulting system in vector form and write $\mathbf{X} = \mathbf{g}(\mathbf{X})$. These systems

are called monotone systems of polynomial equations and have been investigated by Etessami and Yannakakis (2009) and Kiefer et al. (2007). The sought solution, that is, the partition function, is the least fixed point solution of $\mathbf{X} = \mathbf{g}(\mathbf{X})$.

For practical reasons, the set of nonterminals of a grammar is usually divided into maximal subsets of mutually recursive nonterminals, that is, for each A and B in such a subset, we have $A \stackrel{*}{\Rightarrow} uB\alpha$ and $B \stackrel{*}{\Rightarrow} vA\beta$, for some u, v, α, β. This corresponds to a **strongly connected component** if we see the connection between the left-hand side of a rule and a nonterminal member in its right-hand side as an edge in a directed graph. For each strongly connected component, there is a separate system of equations of the form $\mathbf{X} = \mathbf{g}(\mathbf{X})$. Such systems can be solved one by one, in a bottom-up order. That is, if one strongly connected component contains nonterminal A, and another contains nonterminal B, where $A \stackrel{*}{\Rightarrow} uB\alpha$ for some u, α, then the system for the latter component must be solved first.

The solution for a system of equations such as those described above can be irrational and non-expressible by radicals, even if we assume that all the probabilities of the rules in the input PCFG are rational numbers, as observed by Etessami and Yannakakis (2009). Nonetheless, the partition function can still be approximated to any degree of precision by iterative computation of the relation in (4), as done for instance by Stolcke (1995) and by Abney et al. (1999). This corresponds to the so-called fixed-point iteration method, which is well-known in the numerical calculus literature and is frequently applied to systems of non-linear equations because it can be easily implemented.

When a number of standard conditions are met, each iteration of (4) adds a fixed number of bits to the precision of the solution; see Kelley (1995, Chapter 4). Since each iteration can easily be implemented to run in polynomial time, this means that we can approximate the partition function of a PCFG in polynomial time in the size of the PCFG itself and in the number of bits of the desired precision.

In practical applications where large PCFGs are empirically estimated from data sets, the standard conditions mentioned above for the polynomial time approximation of the partition function are usually met. However, there are some degenerate cases for which these standard conditions do not hold, resulting in exponential time behaviour of the fixed-point iteration method. This has been firstly observed in (Etessami and Yannakakis, 2005).

An alternative iterative algorithm for the approximation of the partition function has been proposed by Etessami and Yannakakis (2009), based on Newton's method for the solution of non-linear systems of equations. From a theoretical perspective, Kiefer et al. (2007) have shown that, after a certain number of initial iterations, Newton's method adds a fixed number of bits to the precision of the approximated solution, even in the above mentioned cases in which the fixed-point iteration method shows exponential time behaviour. However, these authors also show that, in some degenerate cases, the number of iterations needed to compute the first bit of the solution can be at least exponential in the size of the system.

Experiments with Newton's method for the approximation of the partition functions of PCFGs have been carried out in several application-oriented settings, by Wojtczak and Etessami (2007) and by Nederhof and Satta (2008), showing considerable improvements over the fixed-point iteration method.

3 Intersection of PCFG and FA

It was shown by Bar-Hillel et al. (1964) that context-free languages are closed under intersection with regular languages. Their proof relied on the construction of a new CFG out of an input CFG and an input finite automaton. Here we extend that construction by letting the input grammar be a probabilistic CFG. We refer the reader to (Nederhof and Satta, 2003) for more details.

To avoid a number of technical complications, we assume the finite automaton has no epsilon transitions, and has only one final state. In the context of our use of this construction in the following sections, these restrictions are without loss of generality. Thus, a finite automaton (FA) $\mathcal{M}$ is represented by a 5-tuple $(\Sigma, Q, q_0, q_f, \Delta)$, where Σ and Q are two finite sets of **terminals** and **states**, respectively, q_0 is the **initial** state, q_f is the **final** state, and Δ is a finite set of **transitions**, each of the form $s \stackrel{a}{\mapsto} t$, where $s, t \in Q$ and $a \in \Sigma$.

A **complete computation** of $\mathcal{M}$ accepting string

$w = a_1 \cdots a_n$ is a sequence $c = \tau_1 \cdots \tau_n$ of transitions such that $\tau_i = (s_{i-1} \overset{a_i}{\mapsto} s_i)$ for each i ($1 \leq i \leq n$), for some $s_0, s_1, \ldots, s_n$ with $s_0 = q_0$ and $s_n = q_f$. The language of all strings accepted by $\mathcal{M}$ is denoted by $L(\mathcal{M})$. A FA is **unambiguous** if at most one complete computation exists for each accepted string. A FA is **deterministic** if there is at most one transition $s \overset{a}{\mapsto} t$ for each s and a.

For a FA $\mathcal{M}$ as above and a PCFG $\mathcal{G} = (\Sigma, N, S, R, p)$ with the same set of terminals, we construct a new PCFG $\mathcal{G}' = (\Sigma, N', S', R', p')$, where $N' = Q \times (\Sigma \cup N) \times Q$, $S' = (q_0, S, q_f)$, and R' is the set of rules that is obtained as follows.

- For each $A \rightarrow X_1 \cdots X_m$ in R and each sequence $s_0, \ldots, s_m$ with $s_i \in Q$, $0 \leq i \leq m$, and $m \geq 0$, let $(s_0, A, s_m) \rightarrow (s_0, X_1, s_1) \cdots (s_{m-1}, X_m, s_m)$ be in R'; if $m = 0$, the new rule is of the form $(s_0, A, s_0) \rightarrow \epsilon$. Function p' assigns the same probability to the new rule as p assigned to the original rule.

- For each $s \overset{a}{\mapsto} t$ in Δ, let $(s, a, t) \rightarrow a$ be in R'. Function p' assigns probability 1 to this rule.

Intuitively, a rule of $\mathcal{G}'$ is either constructed out of a rule of $\mathcal{G}$ or out of a transition of $\mathcal{M}$. On the basis of this correspondence between rules and transitions of $\mathcal{G}'$, $\mathcal{G}$ and $\mathcal{M}$, it is not difficult to see that each derivation d' in $\mathcal{G}'$ deriving string w corresponds to a unique derivation d in $\mathcal{G}$ deriving the same string and a unique computation c in $\mathcal{M}$ accepting the same string. Conversely, if there is a derivation d in $\mathcal{G}$ deriving string w, and some computation c in $\mathcal{M}$ accepting the same string, then the pair of d and c corresponds to a unique derivation d' in $\mathcal{G}'$ deriving the same string w. Furthermore, the probabilities of d and d' are equal, by definition of p'.

Let us now assume that each string w is accepted by at most one computation, i.e. $\mathcal{M}$ is unambiguous. If a string w is accepted by $\mathcal{M}$, then there are as many derivations deriving w in $\mathcal{G}'$ as there are in $\mathcal{G}$. If w is not accepted by $\mathcal{M}$, then there are zero derivations deriving w in $\mathcal{G}'$. Consequently:

$$\sum_{\substack{d', w: \\ S' \overset{d'}{\Rightarrow}_{\mathcal{G}'} w}} p'(d') \quad = \quad \sum_{\substack{d, w: \\ S \overset{d}{\Rightarrow}_{\mathcal{G}} w \wedge w \in L(\mathcal{M})}} p(d),$$

or more succinctly:

$$\sum_{w} p'(w) \quad = \quad \sum_{w \in L(\mathcal{M})} p(w).$$

Note that the above construction of $\mathcal{G}'$ is exponential in the largest value of m in any rule from $\mathcal{G}$. For this reason, $\mathcal{G}$ is usually brought in binary form before the intersection, i.e. the input grammar is transformed to let each right-hand side have at most two members. Such a transformation can be realized in linear time in the size of the grammar. We will return to this issue in Section 7.

4 Obtaining unambiguous FAs

In the previous section, we explained that unambiguous finite automata have special properties with respect to the grammar $\mathcal{G}'$ that we may construct out of a FA $\mathcal{M}$ and a PCFG $\mathcal{G}$. In this section we discuss how unambiguity can be obtained for the special case of finite automata accepting the language of all strings with given infix $w \in \Sigma^*$:

$$L_{infix}(w) \quad = \quad \{xwy \mid x, y \in \Sigma^*\}.$$

Any deterministic automaton is also unambiguous. Furthermore, there seem to be no practical algorithms that turn FAs into equivalent unambiguous FAs other than the algorithms that also determinize them. Therefore, we will henceforth concentrate on deterministic rather than unambiguous automata.

Given a string $w = a_1 \cdots a_n$, a finite automaton accepting $L_{infix}(w)$ can be straightforwardly constructed. This automaton has states $s_0, \ldots, s_n$, transitions $s_0 \overset{a}{\mapsto} s_0$ and $s_n \overset{a}{\mapsto} s_n$ for each $a \in \Sigma$, and transition $s_{i-1} \overset{a_i}{\mapsto} s_i$ for each i ($1 \leq i \leq n$). The initial state is s_0 and the final state is s_n. Clearly, there is nondeterminism in state s_0.

One way to make this automaton deterministic is to apply the general algorithm of determinization of finite automata; see e.g. (Aho and Ullman, 1972). This algorithm is exponential for general FAs. An alternative approach is to construct a deterministic finite automaton directly from w, in line with the Knuth-Morris-Pratt algorithm (Knuth et al., 1977; Gusfield, 1997). Both approaches result in the same deterministic FA, which we denote by $\mathcal{I}_w$. However, the latter approach is easier to implement in such a

way that the time complexity of constructing the automaton is linear in $|w|$.

The automaton $\mathcal{I}_w$ is described as follows. There are $n+1$ states $t_0, \ldots, t_n$, where as before n is the length of w. The initial state is t_0 and the final state is t_n. The intuition is that $\mathcal{I}_w$ reads a string $x = b_1 \cdots b_m$ from left to right, and when it has read the prefix $b_1 \cdots b_j$ ($0 \leq j \leq m$), it is in state t_i ($0 \leq i < n$) if and only if $a_1 \cdots a_i$ is the longest prefix of w that is also a suffix of $b_1 \cdots b_j$. If the automaton is in state t_n, then this means that w is an infix of $b_1 \cdots b_j$.

In more detail, for each i ($1 \leq i \leq n$) and each $a \in \Sigma$, there is a transition $t_{i-1} \overset{a}{\mapsto} t_j$, where j is the length of the longest string that is both a prefix of w and a suffix of $a_1 \cdots a_{i-1}a$. If $a = a_i$, then clearly $j = i$, and otherwise $j < i$. To ensure that we remain in the final state once an occurrence of infix w has been found, we also add transitions $t_n \overset{a}{\mapsto} t_n$ for each $a \in \Sigma$. This construction is illustrated in Figure 1.

5 Infix probability

With the material developed in the previous sections, the problem of computing the infix probabilities can be effectively solved. Our goal is to compute for given infix $w \in \Sigma^*$ and PCFG $\mathcal{G} = (\Sigma, N, S, R, p)$:

$$\sigma_{\mathit{infix}}(w, \mathcal{G}) = \sum_{z \in L_{\mathit{infix}}(w)} p(z).$$

In Section 4 we have shown the construction of finite automaton $\mathcal{I}_w$ accepting $L_{\mathit{infix}}(w)$, by which we obtain:

$$\sigma_{\mathit{infix}}(w, \mathcal{G}) = \sum_{z \in L(\mathcal{I}_w)} p(z).$$

As $\mathcal{I}_w$ is deterministic and therefore unambiguous, the results from Section 3 apply and if $\mathcal{G}' = (\Sigma, N', S', R', p')$ is the PCFG constructed out of $\mathcal{G}$ and $\mathcal{I}_w$ then:

$$\sigma_{\mathit{infix}}(w, \mathcal{G}) = \sum_{z} p'(z).$$

Finally, we can compute the above sum by applying the iterative method discussed in Section 2.

6 Extensions

The approach discussed above allows for a number of generalizations. First, we can replace the infix w by a **sequence** of infixes $w_1, \ldots, w_m$, which have to occur in the given order, one strictly after the other, with arbitrary infixes in between:

$$\sigma_{\mathit{island}}(w_1, \ldots, w_m, \mathcal{G}) = \sum_{x_0, \ldots, x_m \in \Sigma^*} p(x_0 w_1 x_1 \cdots w_m x_m).$$

This problem was discussed before by (Corazza et al., 1991), who mentioned applications in speech recognition. Further applications are found in computational biology, but their discussion is beyond the scope of this paper; see for instance (Apostolico et al., 2005) and references therein. In order to solve the problem, we only need a small addition to the procedures we discussed before. First we construct separate automata $\mathcal{I}_{w_j}$ ($1 \leq j \leq m$) as explained in Section 4. These automata are then composed into a single automaton $\mathcal{I}_{(w_1,\ldots,w_m)}$. In this composition, the outgoing transitions of the final state of $\mathcal{I}_{w_j}$, for each j ($1 \leq j < m$), are removed and that final state is merged with the initial state of the next automaton $\mathcal{I}_{w_{j+1}}$. The initial state of the composed automaton is the initial state of $\mathcal{I}_{w_1}$, and the final state is the final state of $\mathcal{I}_{w_m}$. The time costs of constructing $\mathcal{I}_{(w_1,\ldots,w_m)}$ are linear in the sum of the lengths of the strings w_j.

Another way to generalize the problem is to replace w by a finite set $L = \{w_1, \ldots, w_m\}$. The objective is to compute:

$$\sigma_{\mathit{infix}}(L, \mathcal{G}) = \sum_{w \in L, x, y \in \Sigma^*} p(xwy)$$

Again, this can be solved by first constructing a deterministic FA, which is then intersected with $\mathcal{G}$. This FA can be obtained by determinizing a straightforward nondeterministic FA accepting L, or by directly constructing a deterministic FA along the lines of the Aho-Corasick algorithm (Aho and Corasick, 1975). Construction of the automaton with the latter approach takes linear time.

Further straightforward generalizations involve formalisms such as probabilistic tree adjoining grammars (Schabes, 1992; Resnik, 1992). The technique from Section 3 is also applicable in this case,

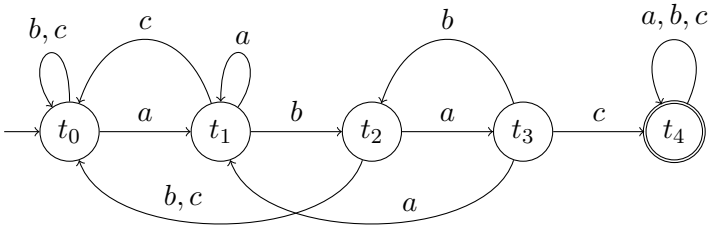


Figure 1: Deterministic automaton that accepts all strings over alphabet $\{a, b, c\}$ with infix *abac*.

as the construction from Bar-Hillel et al. (1964) carries over from context-free grammars to tree adjoining grammars, and more generally to the linear context-free rewriting systems of Vijay-Shanker et al. (1987).

7 Implementation

We have conducted experiments with the computation of infix probabilities. The objective was to identify parts of the computation that have a high time or space demand, and that might be improved. The experiments were run on a desktop with a 3.0 GHz Pentium 4 processor. The implementation language is C++.

The set-up of the experiments is similar to that in (Nederhof and Satta, 2008). A probabilistic context-free grammar was extracted from sections 2-21 of the Penn Treebank version II. Subtrees that generated the empty string were systematically removed. The result was a CFG with 10,035 rules, 28 nonterminals and 36 parts-of-speech. The rule probabilities were determined by maximum likelihood estimation. The grammar was subsequently binarized, to avoid exponential behaviour, as explained in Section 3.

We have considered 10 strings of length 7, randomly generated, assuming each of the parts-of-speech has the same probability. For all prefixes of those strings from length 2 to length 7, we then computed the infix probability. The duration of the full computation, averaged over the 10 strings of length 7, is given in the first row of Table 1.

In order to solve the non-linear systems of equations, we used Broyden's method. It can be seen as an approximation of Newton's method. It requires more iterations, but seems to be faster overall, and more scalable to large problem sizes, due to the avoidance of matrix inversion, which sometimes makes Newton's method prohibitively expensive. In our experiments, Broyden's method was generally faster than Newton's method and much faster than the simple iteration method by the relation in (4). For further details on Broyden's method, we refer the reader to (Kelley, 1995).

The main obstacle to computation for infixes substantially longer than 7 symbols is the memory consumption rather than the running time. This is due to the required square matrices, the dimension of which is the number of nonterminals. The number of nonterminals (of the intersection grammar) naturally grows as the infix becomes longer.

As explained in Section 2, the problem is divided into smaller problems by isolating disjoint sets of mutually recursive nonterminals, or strongly connected components. We found that for the application to the automata discussed in Section 4, there were exactly three strongly connected components that contained more than one element, throughout the experiments. For an infix of length n, these components are:

- C_1, which consists of nonterminals of the form (t_i, A, t_j), where $i < n$ and $j < n$,
- C_2, which consists of nonterminals of the form (t_i, A, t_j), where $i = j = n$, and
- C_3, which consists of nonterminals of the form (t_i, A, t_j), where $i < j = n$.

This can be easily explained by looking at the structure of our automata. See for example Figure 1, with cycles running through states $t_0, \ldots, t_{n-1}$, and cycles through state t_n. Furthermore, the grammar extracted from the Penn Treebank is heavily recursive,

infix length	2	3	4	5	6	7
total running time	1.07	1.95	5.84	11.38	23.93	45.91
Broyden's method for C_1	0.46	0.90	3.42	6.63	12.91	24.38
Broyden's method for C_2	0.08	0.04	0.07	0.04	0.03	0.09
Broyden's method for C_3	0.20	0.36	0.81	1.74	5.30	9.02

Table 1: Running time for infixes from length 2 to length 7. The infixes are prefixes of 10 random strings of length 7, and reported CPU times (in seconds) are averaged over the 10 strings.

so that almost every nonterminal can directly or indirectly call any other.

The strongly connected component C_2 is always the same, consisting of 2402 nonterminals, for each infix of any length. (Note that binarization of the grammar introduced artificial nonterminals.) The last three rows of Table 1 present the time costs of Broyden's method, for the three strongly connected components.

The strongly connected component C_3 happens to correspond to a *linear* system of equations. This is because a rule in the intersection grammar with a left-hand side (t_i, A, t_j), where $i < j = n$, must have a right-hand side of the form (t_i, A', t_j), or of the form $(t_i, A_1, t_k)\ (t_k, A_2, t_j)$, with $k \leq n$. If $k < n$, then only the second member can be in C_3. If $k = n$, only first member can be in C_3. Hence, such a rule corresponds to a linear equation within the system of equations for the entire grammar.

A linear system of equations can be solved analytically, for example by Gaussian elimination, rather than approximated through Newton's method or Broyden's method. This means that the running times in the last row of Table 1 can be reduced by treating C_3 differently from the other strongly connected components. However, the running time for C_1 dominates the total time consumption.

The above investigations were motivated by two questions, namely whether any part of the computation can be precomputed, and second, whether infix probabilities can be computed incrementally, for infixes that are extended to the left or to the right. The first question can be answered affirmatively for C_2, as it is always the same. However, as we can see in Table 1, the computation of C_2 amounts to a small portion of the total time consumption.

The second question can be rephrased more precisely as follows. Suppose we have computed the infix probability of a string w and have kept intermediate results in memory. Can the computation of the infix probability of a string of the form aw or wa, $a \in \Sigma$, be computed by relying on the existing results, so that the computation is substantially faster than if the computation were done from scratch?

Our investigations so far have not found a positive answer to this second question. In particular, the systems of equations for C_1 and C_3 change fundamentally if the infix is extended by one more symbol, which seems to at least make incremental computation very difficult, if not impossible. Note that the algorithms for the computation of prefix probabilities by Jelinek and Lafferty (1991) and Stolcke (1995) do allow incrementality, which contributes to their practical usefulness for speech recognition.

8 Conclusions

We have shown that the problem of infix probabilities for PCFGs can be solved by a construction that intersects a context-free language with a regular language. An important constraint is that the finite automaton that is input to this construction be unambiguous. We have shown that such an automaton can be efficiently constructed. Once the input probabilistic PCFG and the FA have been combined into a new probabilistic CFG, the infix probability can be straightforwardly solved by iterative algorithms. Such algorithms include Newton's method, and Broyden's method, which was used in our experiments. Our discussion ended with an open question about the possibility of incremental computation of infix probabilities.

References

S. Abney, D. McAllester, and F. Pereira. 1999. Relating probabilistic grammars and automata. In *37th Annual*

Meeting of the Association for Computational Linguistics, Proceedings of the Conference, pages 542–549, Maryland, USA, June.

A.V. Aho and M.J. Corasick. 1975. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, June.

A.V. Aho and J.D. Ullman. 1972. *Parsing*, volume 1 of *The Theory of Parsing, Translation and Compiling*. Prentice-Hall, Englewood Cliffs, N.J.

A. Apostolico, M. Comin, and L. Parida. 2005. Conservative extraction of overrepresented extensible motifs. In *Proceedings of Intelligent Systems for Molecular Biology (ISMB05)*.

Y. Bar-Hillel, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*, chapter 9, pages 116–150. Addison-Wesley, Reading, Massachusetts.

T.L. Booth and R.A. Thompson. 1973. Applying probabilistic measures to abstract languages. *IEEE Transactions on Computers*, C-22:442–450.

Z. Chi and S. Geman. 1998. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

A. Corazza, R. De Mori, R. Gretter, and G. Satta. 1991. Computation of probabilities for an island-driven parser. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):936–950.

K. Etessami and M. Yannakakis. 2005. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. In *22nd International Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 340–352, Stuttgart, Germany. Springer-Verlag.

K. Etessami and M. Yannakakis. 2009. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1–66.

A.L.N. Fred. 2000. Computation of substring probabilities in stochastic grammars. In A. Oliveira, editor, *Grammatical Inference: Algorithms and Applications*, volume 1891 of *Lecture Notes in Artificial Intelligence*, pages 103–114. Springer-Verlag.

D. Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, Cambridge.

T.E. Harris. 1963. *The Theory of Branching Processes*. Springer-Verlag, Berlin, Germany.

F. Jelinek and J.D. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323.

C.T. Kelley. 1995. *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

S. Kiefer, M. Luttenberger, and J. Esparza. 2007. On the convergence of Newton's method for monotone systems of polynomial equations. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 217–266.

D.E. Knuth, J.H. Morris, Jr., and V.R. Pratt. 1977. Fast pattern matching in strings. *SIAM Journal on Computing*, 6:323–350.

M.-J. Nederhof and G. Satta. 2003. Probabilistic parsing as intersection. In *8th International Workshop on Parsing Technologies*, pages 137–148, LORIA, Nancy, France, April.

M.-J. Nederhof and G. Satta. 2008. Computing partition functions of PCFGs. *Research on Language and Computation*, 6(2):139–162.

E. Persoon and K.S. Fu. 1975. Sequential classification of strings generated by SCFG's. *International Journal of Computer and Information Sciences*, 4(3):205–217.

P. Resnik. 1992. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proc. of the fifteenth International Conference on Computational Linguistics*, Nantes, August, pages 418–424.

J.-A. Sánchez and J.-M. Benedí. 1997. Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1052–1055, September.

Y. Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proc. of the fifteenth International Conference on Computational Linguistics*, Nantes, August, pages 426–432.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):167–201.

K. Vijay-Shanker, D.J. Weir, and A.K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 104–111, Stanford, California, USA, July.

D. Wojtczak and K. Etessami. 2007. PReMo: an analyzer for Probabilistic Recursive Models. In *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference*, volume 4424 of *Lecture Notes in Computer Science*, pages 66–71, Braga, Portugal. Springer-Verlag.

Parse Correction with Specialized Models for Difficult Attachment Types

Enrique Henestroza Anguiano and Marie Candito
Alpage (Université Paris Diderot / INRIA)
Paris, France
henestro@inria.fr, marie.candito@linguist.jussieu.fr

Abstract

This paper develops a framework for syntactic dependency parse correction. Dependencies in an input parse tree are revised by selecting, for a given dependent, the best governor from within a small set of candidates. We use a discriminative linear ranking model to select the best governor from a group of candidates for a dependent, and our model includes a rich feature set that encodes syntactic structure in the input parse tree. The parse correction framework is parser-agnostic, and can correct attachments using either a generic model or specialized models tailored to difficult attachment types like coordination and pp-attachment. Our experiments show that parse correction, combining a generic model with specialized models for difficult attachment types, can successfully improve the quality of predicted parse trees output by several representative state-of-the-art dependency parsers for French.

1 Introduction

In syntactic dependency parse correction, attachments in an input parse tree are revised by selecting, for a given dependent, the best governor from within a small set of candidates. The motivation behind parse correction is that attachment decisions, especially traditionally difficult ones like pp-attachment and coordination, may require substantial contextual information in order to be made accurately. Because syntactic dependency parsers predict the parse tree for an entire sentence, they may not be able to take into account sufficient context when making attachment decisions, due to computational complexity. Assuming nonetheless that a predicted parse tree is mostly accurate, parse correction can revise difficult attachments by using the predicted tree's syntactic structure to restrict the set of candidate governors and extract a rich set of features to help select among them. Parse correction is also appealing because it is *parser-agnostic*: it can be trained to correct the output of any dependency parser.

In Section 2 we discuss work related to parse correction, pp-attachment and coordination resolution. In Section 3 we discuss dependency structure and various statistical dependency parsing approaches. In Section 4 we introduce the parse correction framework, and Section 5 describes the features and learning model used in our implementation. In Section 6 we present experiments in which parse correction revises the predicted parse trees of four state-of-the-art dependency parsers for French. We provide concluding remarks in Section 7.

2 Related Work

Previous research directly concerning parse correction includes that of Attardi and Ciaramita (2007), working on English and Swedish, who use an approach that considers a fixed set of revision rules: each rule describes movements in the parse tree leading from a dependent's original governor to a new governor, and a classifier is trained to select the correct revision rule for a given dependent. One drawback of this approach is that the classes lack semantic coherence: a sequence of movements does not necessarily have the same meaning across differ-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1222–1233,
Edinburgh, Scotland, UK, July 27–31, 2011.

ent syntactic trees. Hall and Novák (2005), working on Czech, define a neighborhood of candidate governors centered around the original governor of a dependent, and a Maximum Entropy model determines the probability of each candidate-dependent attachment. We follow primarily from their work in our use of neighborhoods to delimit the set of candidate governors. Our main contributions are: specialized corrective models for difficult attachment types (coordination and pp-attachment) in addition to a general corrective model; more sophisticated features, feature combinations, and feature selection; and a ranking model trained directly to select the true governor from among a set of candidates.

There has also been other work on techniques similar to parse correction. Attardi and Dell'Orletta (2009) investigate *reverse revision*: a left-to-right transition-based model is first used to parse a sentence, then a right-to-left transition-based model is run with additional features taken from the left-to-right model's predicted parse. This approach leads to improved parsing results on a number of languages. While their approach is similar to parse correction in that it uses a predicted parse to inform a subsequent processing step, this information is used to improve a second parser rather than a model for correcting errors. McDonald and Pereira (2006) consider a method for recovering non-projective attachments from a graph representation of a sentence, in which an optimal projective parse tree has been identified. The parse tree's edges are allowed to be rearranged in ways that introduce non-projectivity in order to increase its overall score. This rearrangement approach resembles parse correction because it is a second step that can revise attachments made in the first step, but it differs in a number of ways: it is dependent on a graph-based parsing approach, it does not model errors made by the parser, and it can only output non-projective variants of the predicted parse tree.

As a process that revises the output of a syntactic parser, parse reranking is also similar to parse correction. A well-studied subject (e.g. the work of Charniak and Johnson (2005) and of Collins and Koo (2005)), parse reranking is concerned with the reordering of n-best ranked parse trees output by a syntactic parser. Parse correction has a number of advantages compared to reranking: it can be used with parsers that do not output n-best ranked parses, it can be easily restricted to specific attachment types, and its output space of parse trees is not limited to those appearing in an n-best list. However, parse reranking has the advantage of selecting the globally optimal parse for a sentence from an n-best list, while parse correction makes only locally optimal revisions in the predicted parse for a sentence.

2.1 Difficult Attachment Types

Research on pp-attachment traditionally formulates the problem in isolation, as in the work of Pantel and Lin (2000) and of Olteanu and Moldovan (2005). Examples consist of tuples of the form (v, n_1, p, n_2), where either v or n_1 is the true governor of the pp comprising p and n_2, and the task is to choose between v and n_1. Recently, Atterer and Schütze (2007) have criticized this formulation as unrealistic because it uses an oracle to select candidate governors, and they find that successful approaches for the isolated problem perform no better than state-of-the-art parsers on pp-attachment when evaluated on full sentences. With parse correction, candidate governors are identified automatically with no (v, n_1, p, n_2) restriction, and for several representative parsers we find that parse correction improves pp-attachment performance.

Research on coordination resolution has also often formulated the problem in isolation. Resnik (1999) uses semantic similarity to resolve noun-phrase coordination of the form (n_1, cc, n_2, n_3), where the coordinating conjunction cc coordinates either the heads n_1 and n_2 or the heads n_1 and n_3. The same criticism as the one made by Atterer and Schütze (2007) for pp-attachment might be applied to this approach to coordination resolution. In another formulation, the input consists of a raw sentence, and coordination structure is then detected and disambiguated using discriminative learning models (Shimbo and Hara, 2007) or coordination-specific parsers (Hara et al., 2009). Finally, other work has focused on introducing specialized features for coordination into existing syntactic parsing models (Hogan, 2007). Our approach is novel with respect to previous work by directly modeling the correction of coordination errors made by general-purpose dependency parsers.

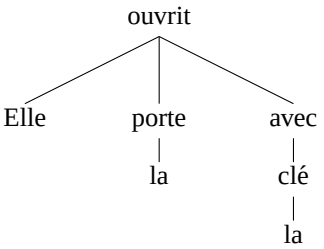


Figure 1: An unlabeled dependency tree for: *Elle ouvrit la porte avec la clé.* (*She opened the door with the key*).

3 Dependency Parsing

Dependency syntax involves the representation of syntactic information for a sentence in the form a directed graph, whose edges encode word-to-word relationships. An edge from a *governor* to a *dependent* indicates, roughly, that the presence of the dependent is syntactically legitimated by the governor. An important property of dependency syntax is that each word, except for the root of the sentence, has exactly one governor; dependency syntax is thus represented by trees. Figure 1 shows an example of an unlabeled dependency tree.[1] For languages like English or French, most sentences can be represented with a *projective* dependency tree: for any edge from word g to word d, g dominates any intervening word between g and d.

Dependency trees are appealing syntactic representations, closer than constituency trees to the semantic representations useful for NLP applications. This is true even with the projectivity requirement, which occasionally creates syntax-semantics mismatches. Dependency trees have recently seen a surge of interest, particularly with the introduction of supervised models for dependency parsing using linear classifiers. Such parsers fall into two main categories: transition-based parsing and graph-based parsing. Additionally, an alternative method for obtaining the dependency parse for a sentence is to parse the sentence with a constituency-based parser and then use an automatic process to convert the output into dependency structure.

[1]Edges are generally labeled with the surface grammatical function that the dependent bears with respect to its governor. In this paper we focus on unlabeled dependency parsing, setting aside labeling as a separate task.

3.1 Transition-Based Parsing

In transition-based dependency parsing, whose seminal works are that of Yamada and Matsumoto (2003) and Nivre (2003), the parsing process applies a sequence of incremental actions, which typically manipulate a buffer position in the sentence and a stack for built sub-structures. Actions are of the type "*read word from buffer*", "*build a dependency from node on top of the stack to node that begins the buffer*", etc. In a greedy version of this process, the action to apply at each step is deterministically chosen to be the best-scoring action according to a classifier, which is trained on a dependency treebank converted into sequences of actions. The strengths of this framework are $O(n)$ time complexity and a lack of restrictions on the locality of features. A major drawback is its greedy behavior: it can potentially make difficult attachment decisions early in the processing of a sentence, without being able to reconsider them when more information becomes available. Beamed versions of the algorithm (Johansson and Nugues, 2006) partially address this problem, but still do not provide a global optimization for selecting the output parse tree.

3.2 Graph-Based Parsing

In graph-based dependency parsing, whose seminal work is that of McDonald et al. (2005), the parsing process selects the globally optimal parse tree from a graph containing attachments (directed edges) between each pair of words (nodes) in a sentence. It finds the k-best scoring parse trees, both during training and at parse time, where the score of a tree is the sum of the scores of its *factors* (consisting of one or more linked edges). While large factors are desirable for capturing sophisticated linguistic constraints, they come at the cost of time complexity: for the projective case, adaptations of Eisner's algorithm (Eisner, 1996) are $O(n^3)$ for 1-edge factors (McDonald et al., 2005) or sibling 2-edge factors (McDonald and Pereira, 2006), and $O(n^4)$ for general 2-edge factors (Carreras, 2007) or 3-edge factors (Koo and Collins, 2010).

3.3 Constituency-Based Parsing

Beyond the two main approaches to dependency parsing, there is also the approach of constituency-

based parsing followed by a conversion step to dependency structure. We use the three-step parsing architecture previously tested for French by Candito et al. (2010a): (i) A constituency parse tree is output by the BerkeleyParser, which has been trained to learn a probabilistic context-free grammar with latent annotations (Petrov et al., 2006) that has parsing time complexity $O(n^3)$ (Matsuzaki et al., 2005); (ii) A functional role labeler using a Maximum Entropy model adds functional annotations to links between a verb and its dependents; (iii) Constituency trees are automatically converted into projective dependency trees, with remaining unlabeled dependencies assigned labels using a rule-based approach.

3.4 Baseline Parsers

In this paper, we use the following baseline parsers: MaltParser (Nivre et al., 2007) for transition-based parsing; MSTParser (McDonald et al., 2005) (with sibling 2-edge factors) and BohnetParser (Bohnet, 2010) (with general 2-edge factors) for graph-based parsing; and BerkeleyParser (Petrov et al., 2006) for constituency-based parsing.

For MaltParser and MSTParser, we use the best settings from a benchmarking of parsers for French (Candito et al., 2010b), except that we remove unsupervised word clusters as features. The parsing models are thus trained using features including predicted part-of-speech tags, lemmas and morphological features. For BohnetParser, we trained a new model using these same predicted features. For BerkelyParser, which was included in the benchmarking experiments, we trained a model using the so-called "desinflection" process that addresses data sparseness due to morphological variation: both at training and parsing time, terminal symbols are word forms in which redundant morphological suffixes are removed, provided the original part-of-speech ambiguities are kept (Candito et al., 2010b).

All models are trained on the French Treebank (FTB) (Abeillé and Barrier, 2004), consisting of 12,351 sentences from the *Le Monde* newspaper, either "desinflected" for the BerkeleyParser, or converted to projective dependency trees (Candito et al., 2010a) for the three dependency-native parsers.[2] For the dependency-native models, features include predicted part-of-speech (POS) tags from the MElt tagger (Denis and Sagot, 2009), as well as predicted lemmas and morphological features from the Le*fff* lexicon (Sagot, 2010). These models constitute the state-of-the-art for French dependency parsing: unlabeled attachment scores (UAS) on the FTB test set are 89.78% for MaltParser, 91.04% for MSTParser, 91.78% for BohnetParser, and 90.73% for BerkeleyParser.

INPUT: Predicted parse tree T

LOOP: For each chosen dependent $d \in D$

- Identify candidates C_d from T
- Predict $\hat{c} = \underset{c \in C_d}{\operatorname{argmax}}\ S(c, d, T)$
- Update $T\{gov(d) \leftarrow \hat{c}\}$

OUTPUT: Corrected version of parse tree T

Figure 2: The parse correction algorithm.

4 Parse Correction

The parse correction algorithm is a post-processing step to dependency parsing, where attachments from the predicted parse tree of a sentence are corrected by considering alternative candidate governors for each dependent. This process can be useful for attachments made too early in transition-based parsing, or with features that are too local in MST-based parsing.

The input is the predicted parse T of a sentence. From T a set D of dependent nodes are chosen for attachment correction. For each $d \in D$ in left-to-right sentence order, a set C_d of candidate governors from T is identified, and then the highest scoring $c \in C_d$, using a function $S(c, d, T)$, is assigned as the new governor of d in T. Pseudo-code for parse correction is shown in Figure 2.[3]

[2] The projectivity constraint is linguistically valid for most French parses: the authors report $< 2\%$ non-projective edges in a hand-corrected subset of the converted FTB.

[3] Contrary to Hall and Novák (2005), our iterative algorithm (along with the fact that C_d never includes nodes that are dominated by d) ensures that corrected structures are trees, so it does not require additional processing to eliminate cycles and preserve connectivity.

4.1 Choosing Dependents

Various criteria may be used to choose the set D of dependents to correct. In the work of Hall and Novák (2005) and of Attardi and Ciaramita (2007), D contains all nodes in the input parse tree. However, one advantage of parse correction is its ability to focus on specific attachment types, so an additional criterion for choosing dependents is to look separately at those dependents that correspond to difficult attachment types.

Analyzing errors made by the dependency parsers introduced in Section 3 on the development set of the FTB, we observe that two major sources of error across different parsers are coordination and pp-attachment. Coordination accounts for around 10% of incorrect attachments and has an error rate ranging from $30 - 40\%$, while pp-attachment accounts for around 30% of incorrect attachments and has an error rate of around 15%.

In this paper, we pay special attention to coordination and pp-attachment. Given the FTB annotation scheme, coordination can be corrected by changing the governor (first conjunct) of the coordinating conjunction that governs the second conjunct, and pp-attachment can be corrected by changing the governor of the preposition that heads the pp.[4] We thus train specialized corrective models for when the dependents are coordinating conjunctions and prepositions, in addition to a generic corrective model that can be applied to any dependent.[5]

4.2 Identifying Candidate Governors

The set of candidate governors C_d for a dependent d can be chosen in different ways. One method is to let every other node in T be a candidate governor for d. However, parser error analysis has shown that errors often occur in local contexts. Hall and Novák (2005) define a neighborhood as a set of nodes $N_m(d)$ around the original predicted governor c_o of d, where $N_m(d)$ includes all nodes in the parse tree T within graph distance m of d that pass through c_o. They find that around $2/3$ of the incorrect attachments in the output of Czech parses can be corrected by selecting the best governor from within $N_3(d)$. Similarly, in oracle experiments reported in section 6, we find that around $1/2$ of coordination and pp-attachments in the output of French parses can be corrected by selecting the best governor from within $N_3(d)$. We thus use neighborhoods to delimit the set of candidate governors.

While one can simply assign $C_d \leftarrow N_m(d)$, we add additional restrictions. First, in order to preserve projectivity within T, we keep in C_d only those c such that the update $T\{gov(d) \leftarrow c\}$ would result in a projective tree.[6] Additionally, we discard candidates with certain POS categories that are very unlikely to be governors: clitics and punctuation are always discarded, while determiners are discarded if the dependent is a preposition.

4.3 Scoring Candidate Governors

A new governor $\hat{c}$ for a dependent d is predicted by selecting the highest scoring candidate $c \in C_d$ according to a function $S(c, d, T)$, which takes into account features over c, d, and the parse tree T. We use a linear model for our scoring function, which allows for relatively fast training and prediction. Our scoring function uses a weight vector $\vec{w} \in \mathbb{F}$, where $\mathbb{F}$ is the feature space for dependents we wish to correct (either generic, or specialized for prepositions or for coordinating conjunction), as well as the mapping $\Phi : \mathbb{C}\times\mathbb{D}\times\mathbb{T} \rightarrow \mathbb{F}$ from combinations of candidate $c \in \mathbb{C}$, dependent $d \in \mathbb{D}$, and parse tree $T \in \mathbb{T}$, to vectors in the feature space $\mathbb{F}$. The scoring function returns the inner product of $\vec{w}$ and $\Phi(c, d, T)$:

$$S(c, d, T) = \vec{w} \cdot \Phi(c, d, T) \quad (1)$$

4.4 Algorithm Complexity

The time complexity of our algorithm is $O(n)$ in the length n of the input sentence, which is consistent with past work on parse correction by Hall and Novák (2005) and by Attardi and Ciaramita (2007).

[4] The FTB handles pp-attachment in a typical fashion, but coordination may be handled differently by other schemes (e.g. the coordinating conjunction governs both conjuncts).

[5] In our experiments, we never revise punctuation and clitic dependents. Since punctuation attachments mostly carry little meaning, they are often annotated inconsistently and ignored in parsing evaluations (including ours). Clitics are not revised because they have a very low attachment error rate (2%).

[6] We also keep candidates that would lead to a non-projective tree, as long as it would be projective if we ignored punctuation. This relaxation of the projectivity constraint leads to better oracle scores while retaining the key linguistic properties of projectivity.

Attachments for up to n dependents in a sentence are deterministically corrected in one pass. For each such dependent d, the algorithm uses a linear model to select a new governor after extracting features for a local set of candidate governors C_d, whose size does not dependent on n in the average case.[7] Locality in candidate governor identification and feature extraction preserves linear time complexity in the overall algorithm.

5 Model Learning

We now discuss our training setup, features, and learning approach for obtaining the weight vector $\vec{w}$.

5.1 Training Setup

The parse correction training set pairs gold parse trees with corresponding predicted parse trees output by a syntactic parser, and it is obtained using a jackknifing procedure to automatically parse the gold-annotated training section of a dependency treebank with a syntactic dependency parser.

We extract separate training sets for each type of dependent we wish to correct (generic, prepositions, coordinating conjunctions). Given p, then for each token d we wish to correct in a sentence in the training section, we note its true governor g_d in the gold parse tree of the sentence, identify a set of candidate governors C_d in the predicted parse T, and get feature vectors $\{\Phi(c, d, T) : c \in C_d\}$.

5.2 Feature Space

In order to learn an effective scoring function, we use a rich feature space $\mathbb{F}$ that encodes syntactic context surrounding a candidate-dependent pair (c, d) within a parse tree T. Our primary features are indicator functions for realizations of linguistic or tree-based feature classes.[8] From these primary features we generate more complex feature combinations of length up to P, which are then added to $\mathbb{F}$. Each combo represents a set of one or more primary features, and is an indicator function that fires if and only if all of its members do.

[7] Degenerate parse trees (e.g. flat trees) could lead to cases where $|C_d|$=n, but for linguistically coherent parse trees $|C_d|$ is rather $O(k^m)$, where k is the average -*arity* of syntactic parse trees and m is the neighborhood distance used.

[8] For instance, there is a binary feature that is 1 if feature class "POS of c" takes on the value "verb", and 0 otherwise.

5.2.1 Primary Feature Classes

The primary feature classes we use are listed below, grouped into categories corresponding to their use in different corrective models (d_{obj} is the object of the dependent, c_{gov} is the governor of the candidate, and c_{d-1} and c_{d+1} are the closest dependents of c linearly to the left and right, respectively, of d).

Generic features (always included)

- POS, lemma, and number of dependents of c
- POS and dependency label of c_{d-1}
- POS and dependency label of c_{d+1}
- POS of c_{gov}
- POS and lemma of d
- POS of d_{obj} and whether d_{obj} has a determiner
- Whether c is the predicted governor of d
- Binned linear distance between c and d
- Linear direction of c with respect to d
- POS sequence for nodes on path from c to d
- Graph distance between c and d
- Whether there is punctuation between c and d

Features exclusive to coordination

Whether d would coordinate two conjuncts that:

- Have the same POS
- Have the same word form
- Have number agreement
- Are both nouns with the same cardinality
- Are both proper nouns or both common nouns
- Are both prepositions with the same word form
- Are both prepositions with object of same POS

Features exclusive to pp-attachment

- Whether d immediately follows a punctuation
- Whether d heads a pp likely to be the agent of a passive verb
- If c is a coordinating conjunction, then whether c would coordinate two prepositions with the same word form, and whether there is at least one open-category word linearly between c and d (in which case c is an unlikely governor)

– If c is linearly after d, then whether there exists a plausible rival candidate to the left of d (implemented as whether there is a noun or adjective linearly before d, without any intervening finite verb)

5.2.2 Feature Selection

Feature combos allow our models to effectively sidestep linearity constraints, at the cost of an exponential increase in the size of the feature space $\mathbb{F}$. In order to accommodate combos, we use feature selection to help reduce the resulting space.

Our first feature selection technique is to apply a frequency threshold: if a feature or a combo appears less than K times among instances in our training set, we remove it from $\mathbb{F}$. In addition to making the feature space more tractable, frequency thresholding makes our scoring function less reliant on rare features and combos.

Following frequency thresholding, we employ an additional technique using conditional entropy (CE) that we term *CE-reduction*. Let Y be a random variable for whether or not an attachment is true, and let A be a random variable for different combos that can appear in an attachment. We calculate the CE of a combo a with respect to Y as follows,

$$H(Y|A{=}a) = -\sum_{y \in Y} p(y|a) \log p(y|a) \quad (2)$$

where the probability $p(y|a)$ is approximated from the training set as $freq(a, y)/freq(a)$, with example balancing used here to account for more false attachments ($Y = 0$) than true ones ($Y = 1$) in our training set. Having calculated the CE of each combo, we remove from $\mathbb{F}$ those combos for which a subset combo (or feature) exists with equal or lesser CE. This eliminates any overly specific combo a when the extra features encoded in a, compared to some subset b, do not help a explain Y any better than b.

5.3 Ranking Model

The ranking setting for learning is used when a model needs to discriminate between mutually exclusive candidates that vary from instance to instance. This is typically used in parse reranking (Charniak and Johnson, 2005), where for each sentence the model must select the correct parse from within an n-best list. Denis and Baldridge (2007) also show that ranking outperforms a binary classification approach to pronoun resolution (using a Maximum Entropy model), where for each pronominal anaphor the model must select the correct antecedent among candidates in a text.[9]

INPUT: Aggressiveness C, rounds R.
INITIALIZE: $\vec{w}_0 \leftarrow (0, ..., 0), \vec{w}_{avg} \leftarrow (0, ..., 0)$
REPEAT: R times
LOOP: For $t = 1, 2, \ldots, |X|$
· Get feature vectors $\{\vec{x}_{t,c} : c \in C_{d_t}\}$
· Get true governor $g_t \in C_{d_t}$
· Let $h_t = \underset{c \in C_{d_t} - \{g_t\}}{\text{argmax}} \ (\vec{w}_{t-1} \cdot \vec{x}_{t,c})$
· Let $m_t = (\vec{w}_{t-1} \cdot \vec{x}_{t,g_t}) - (\vec{w}_{t-1} \cdot \vec{x}_{t,h_t})$
IF: $m_t < 1$
· Let $\tau_t = \min\left\{C\ ,\ \frac{1-m_t}{\|\vec{x}_{t,g_t} - \vec{x}_{t,h_t}\|^2}\right\}$
· Set $\vec{w}_t \leftarrow \vec{w}_{t-1} + \tau_t(\vec{x}_{t,g_t} - \vec{x}_{t,h_t})$
ELSE:
· Set $\vec{w}_t \leftarrow \vec{w}_{t-1}$
· Set $\vec{w}_{avg} \leftarrow \vec{w}_{avg} + \vec{w}_t$
· Set $\vec{w}_0 \leftarrow \vec{w}_{|X|}$
OUTPUT: $\vec{w}_{avg}/(R \cdot |X|)$

Figure 3: Averaged PA-Ranking training algorithm.

In our ranking approach to parse correction (PA-Ranking), the weight vector is trained to select the true governor from a set of candidates C_d for a dependent d. The training set X is defined such that the t^{th} instance is a collection of feature vectors $\{\vec{x}_{t,c} = \Phi(c, d_t, T_t) : c \in C_{d_t}\}$, where C_{d_t} is the candidate set for the dependent d_t within the predicted parse T_t, and the class is the true governor g_t. Instances in which $g_t \notin C_{d_t}$ are discarded.

PA-Ranking training is carried out using a variation of the Passive-Aggressive algorithm (Crammer et al., 2006), which has been adapted to the ranking setting, implemented using the Polka library.[10] For each training iteration t, the margin is defined as

[9] We considered a binary training approach to parse correction in which the model is trained to independently classify candidates as true or false governors, as used by Hall and Novák (2005). However, we found that this approach performs no better (and often worse) than the ranking approach, and is less appropriate from a modeling standpoint.

[10] `http://polka.gforge.inria.fr/`

$m_t = (\vec{w}_{t-1} \cdot \vec{x}_{t,g_t}) - (\vec{w}_{t-1} \cdot \vec{x}_{t,h_t})$, where h_t is the highest scoring incorrect candidate. The algorithm is *passive* because an update to the weight vector is made if and only if $m_t < 1$, either for incorrect predictions ($m_t < 0$) or for correct predictions with insufficient margin ($0 \leq m_t < 1$). The new weight vector $\vec{w}_t$ is as close as possible to $\vec{w}_{t-1}$, subject to the *aggressive* constraint that the new margin be greater than 1. We use weight averaging, so the final output $\vec{w}_{avg}$ is the average over the weight vectors after each training step. Pseudo-code for the training algorithm is shown in Figure 3. The rounds parameter R determines the number of times to run through the training set, and the aggressiveness parameter C sets an upper limit on the update magnitude.

6 Experiments

We present experiments where we applied parse correction to the output of four state-of-the-art dependency parsers for French. We conducted our evaluation on the FTB using the standard training, development (dev), and test splits (containing 9,881, 1,235 and 1,235 sentences, respectively). To train our parse correction models, we generated specialized training sets corresponding to each parser by doing 10-fold jackknifing on the FTB training set (cf. Section 5.1). Each parser was run on the FTB dev and test sets, providing baseline unlabeled attachment score (UAS) results and output parse trees to be corrected.

6.1 Oracles and Neighborhood Size

To determine candidate neighborhood size, we considered an oracle scoring function that always selects the true governor of a dependent if it appears in the set of candidate governors, and otherwise selects the predicted governor. Results for this oracle on the dev set are shown in Table 1. The baseline corresponds to m=1, where the oracle just selects the predicted governor. Incrementing m to 2 and to 3 resulted in substantial gains in oracle UAS, but further incrementing m to 4 resulted in a relatively small additional gain. We found that average candidate set size increases about linearly in m, so we decided to use m=3 in order to have a high UAS upper bound without adding candidates that are very unlikely to be true governors.

		Neighborhood Size (m)			
		Base	2	3	4
Berkeley	Coords	67.2	76.5	82.8	84.8
	Preps	82.9	88.5	92.2	93.2
	Overall	90.1	94.0	96.0	96.5
Bohnet	Coords	70.1	80.6	85.6	87.7
	Preps	85.4	89.4	93.4	94.5
	Overall	91.2	94.4	96.1	96.6
Malt	Coords	60.9	72.2	78.2	80.5
	Preps	82.6	88.1	92.6	93.7
	Overall	89.3	93.2	95.1	95.8
MST	Coords	63.6	73.7	80.7	84.4
	Preps	84.7	89.4	93.4	94.4
	Overall	90.2	93.7	95.6	96.2
MST	Overall	Reranking top-100 parses: 95.4			

Table 1: Parse correction oracle UAS (%) for different neighborhood sizes, by dependent type (coordinating conjunctions, prepositions, or all dependents). Also, a reranking oracle for MSTParser using the top-100 parses.

We also compared the oracle for parse correction with an oracle for parse reranking, in which the parse with the highest UAS for a sentence is selected from the top-100 parses output by MSTParser. We found that for MSTParser, the oracle for parse correction using neighborhood size m=3 (95.6% UAS) is comparable to the oracle for parse reranking using the top-100 parses (95.4% UAS). This is an encouraging result, showing that parse correction is capable of the same improvement as parse reranking without needing to process an n-best list of parses.

6.2 Feature Space Parameters

For the feature space $\mathbb{F}$, we performed a grid search to find good values for the parameters K (frequency threshold), P (combo length), and CE-reduction. We found that P=3 with CE-reduction allowed for the most compactness without sacrificing correction performance, for all of our corrective models. Additionally, K=2 worked well for the coordinating conjunction models, while K=10 worked well for the preposition and generic models. CE-reduction proved useful in greatly reducing the feature space without lowering correction performance: it reduced the size of the coordinating conjunction models from 400k to 65k features each, the preposition models from 400k to 75k features each, and the generic models from 800k to 200k features each.

	Corrective Configuration	UAS (%)		
		Coords	Preps	Overall
Berkeley	Baseline	68.3	83.8	90.73
	Generic	69.4	84.9*	91.13*
	Specialized	71.5*	85.1*	91.23*
Bohnet	Baseline	70.5	86.1	91.78
	Generic	71.2	86.4	91.88
	Specialized	72.7*	86.2	91.88
Malt	Baseline	59.8	83.2	89.78
	Generic	63.2*	84.5*	90.39*
	Specialized	64.0*	85.0*	90.47*
MST	Baseline	60.5	85.9	91.04
	Generic	64.2*	86.2	91.25*
	Specialized	68.0*	86.2	91.36*

Table 2: Coordinating conjunction, preposition, and overall UAS (%) by corrective configuration on the test set. Significant improvements over the baseline starred.

6.3 Corrective Configurations

For our evaluation of parse correction, we compared two different configurations: *generic* (corrects all dependents using the generic model) and *specialized* (corrects coordinating conjunctions and prepositions using their respective specialized models, and corrects other dependents using the generic model). The PA-Ranking aggressiveness parameter C was set to 1 for our experiments, while the rounds parameter R was tuned separately for each corrective model using the dev set. For our final tests, we applied each combination of parser + corrective configuration by sequentially revising all dependents in the output parse that had a relevant POS tag given the corrective configuration. In the FTB test set, this amounted to an evaluation over 5,706 preposition tokens, 801 coordinating conjunction tokens, and 31,404 overall (non-punctuation) tokens.[11]

6.4 Results

Final results for the test set are shown in Table 2. The overall UAS of each parser (except BohnetParser) was significantly improved under both corrective configurations.[12] The *specialized* configuration performed as well as, and in most cases better than, the *generic* configuration, indicating the usefulness of specialized models and features for difficult attachment types. Interestingly, the lower the baseline parser's UAS, the larger the overall improvement from parse correction under the *specialized* configuration: MaltParser had the lowest baseline and the highest error reduction (6.8%), BerkeleyParser had the second-lowest baseline and the second-highest error reduction (5.4%), MSTParser had the third-lowest baseline and the third-highest error reduction (3.6%), and BohnetParser had the highest baseline and the lowest error reduction (1.2%). It may be that the additional errors made by a low-baseline parser, compared to a high-baseline parser, involve relatively simpler attachments that parse correction can better model.

Parse correction achieved significant improvements for coordination resolution under the *specialized* configuration for each parser. MaltParser and MSTParser had very low baseline coordinating conjunction UAS (around 60%), while BerkeleyParser and BohnetParser had higher baselines (around 70%). The highest error reduction was achieved by MSTParser (19.0%), followed by MaltParser (10.4%), BerkeleyParser (10.1%), and finally BohnetParser (7.5%). The result for MSTParser was surprising: although it had the second-highest baseline overall UAS, it shared the lowest baseline coordinating conjunction UAS and had the highest error reduction with parse correction. An explanation for this result is that the annotation scheme for coordination structure in the dependency FTB has the first conjunct governing the coordinating conjunction, which governs the second conjunct. Since MSTParser is limited to sibling 2-edge factors (cf. section 3), it is unable to jointly consider a full coordination structure. BohnetParser, which uses general 2-edge factors, can consider full coordination structures and consequently has a much higher baseline coordinating conjunction UAS than MSTParser.

Parse correction achieved significant but modest improvements in pp-attachment performance under the *specialized* configuration for MaltParser and BerkeleyParser. However, parse correction did not significantly improve pp-attachment performance for MSTParser or BohnetParser, the two parsers that had the highest baseline preposition UAS (around

[11] Since the MElt tagger and BerkeleyParser POS tagging accuracies were around 97%, the sets of tokens considered for revision differed slightly from the sets of tokens (with gold POS tags) used to calculate UAS scores.

[12] We used McNemar's Chi-squared test with $p = 0.05$ for all significance tests.

		Modification Type			
		$w \rightarrow c$	$c \rightarrow w$	$w \rightarrow w$	Mods
Berkeley	Coords	40	14	33	10.9 %
	Preps	118	39	41	3.5 %
	Overall	228	67	104	1.3 %
Bohnet	Coords	32	15	33	10.0 %
	Preps	52	46	32	2.3 %
	Overall	150	121	130	1.1 %
Malt	Coords	55	21	56	16.5 %
	Preps	149	50	76	4.8 %
	Overall	390	172	293	2.4 %
MST	Coords	80	20	51	18.9 %
	Preps	64	45	26	2.4 %
	Overall	183	88	117	1.1 %

Table 3: Breakdown of modifications made under the *specialized* configuration for each parser, by dependent type. $w \rightarrow c$ is wrong-to-correct, $c \rightarrow w$ is correct-to-wrong, $w \rightarrow w$ is wrong-to-wrong, and Mods is the percentage of tokens modified.

86%). These results are a bit disappointing, but they suggest that there may be a performance ceiling for pp-attachment beyond which rich lexical information (syntactic and semantic) or full sentence contexts are needed. For English, the average human performance on pp-attachment for the (v, n_1, p, n_2) problem formulation is just 88.2% when given only the four head-words, but increases to 93.2% when given the full sentence (Ratnaparkhi et al., 1994). If similar levels of human performance exist for French, additional sources of information may be needed to improve pp-attachment performance.

In addition to evaluating UAS improvements for parse correction, we took a closer look at the best corrective configuration (*specialized*) and analyzed the types of attachment modifications made (Table 3). In most cases there were around $2-3$ times as many error-correcting modifications ($w \rightarrow c$) as error-creating modifications ($c \rightarrow w$), and the overall % of tokens modified was very low overall (around 1-2%). Parse correction is thus conservative in the number of modifications made, and rather accurate when it does decide to modify an attachment.

Finally, we compared the running times of the four parsers, as well as that of parse correction, on the test set using a 2.66 GHz Intel Core 2 Duo machine. BerkeleyParser took 600s, BohnetParser took 450s using both cores (800s using a single core), MaltParser took 45s, and MSTParser took 1000s. A rough version of parse correction in the *specialized* configuration took around 200s (for each parser). An interesting result is that parse correction improves MaltParser the most while retaining an overall time complexity of $O(n)$, compared to $O(n^3)$ or higher for the other parsers. This suggests that linear-time transition-based parsing and parse correction could combine to form an attractive system that improves parsing performance while retaining high speed.

7 Conclusion

We have developed a parse correction framework for syntactic dependency parsing that uses specialized models for difficult attachment types. Candidate governors for a given dependent are identified in a neighborhood around the predicted governor, and a scoring function selects the best governor. We used discriminative linear ranking models with features encoding syntactic context, and we tested parse correction on coordination, pp-attachment, and generic dependencies in the outputs of four representative statistical dependency parsers for French. Parse correction achieved improvements in unlabeled attachment score for three out of the four parsers, with MaltParser seeing the greatest improvement. Since both MaltParser and parse correction run in $O(n)$ time, a combined system could prove useful in situations where high parsing speed is required.

Future work on parse correction might focus on developing specialized models for other difficult attachment types, such as verb-phrase attachment (verb dependents account for around 15% of incorrect attachments across all four parsers). Also, selectional preferences and subcategorization frames (from hand-built resources or extracted using distributional methods) could make for useful features in the pp-attachment corrective model; we suspect that richer lexical information is needed in order to increase the currently modest improvements achieved by parse correction on pp-attachment.

Acknowledgments

We would like to thank Pascal Denis for his help using the Polka library, and Alexis Nasr for his advice and comments. This work was partially funded by the ANR project Sequoia ANR-08-EMER-013.

References

A. Abeillé and N. Barrier. 2004. Enriching a French treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, Lisbon, Portugal, May.

G. Attardi and M. Ciaramita. 2007. Tree revision learning for dependency parsing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 388–395, Rochester, New York, April.

G. Attardi and F. Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 261–264, Boulder, Colorado, June.

M. Atterer and H. Schütze. 2007. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476.

B. Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97, Beijing, China, August.

M. Candito, B. Crabbé, and P. Denis. 2010a. Statistical French dependency parsing: Treebank conversion and first results. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, Valetta, Malta, May.

M. Candito, J. Nivre, P. Denis, and E. Henestroza Anguiano. 2010b. Benchmarking of statistical dependency parsers for French. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 108–116, Beijing, China, August.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 957–961, Prague, Czech Republic, June.

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

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

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

P. Denis and J. Baldridge. 2007. A ranking approach to pronoun resolution. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, pages 1588–1593, Hyderabad, India, January.

P. Denis and B. Sagot. 2009. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*, Hong Kong, China, December.

J.M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345, Santa Cruz, California, August.

K. Hall and V. Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies*, pages 42–52, Vancouver, British Columbia, October.

K. Hara, M. Shimbo, H. Okuma, and Y. Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 967–975, Suntec, Singapore, August.

D. Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, number 1, page 680, Prague, Czech Republic, June.

R. Johansson and P. Nugues. 2006. Investigating multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 206–210, New York City, New York, June.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden, July.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 75–82, Ann Arbor, Michigan, June.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy, April.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98, Ann Arbor, Michigan, June.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 149–160, Nancy, France, April.

M. Olteanu and D. Moldovan. 2005. PP-attachment disambiguation using large context. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 273–280, Vancouver, British Columbia, October.

P. Pantel and D. Lin. 2000. An unsupervised approach to prepositional phrase attachment using contextually similar words. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, volume 38, pages 101–108, Hong Kong, October.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July.

A. Ratnaparkhi, J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the Workshop on Human Language Technology*, pages 250–255, Plainsboro, New Jersey, March.

P. Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11(95):130.

B. Sagot. 2010. The Lefff, a freely available, accurate and large-coverage lexicon for French. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, Valetta, Malta, May.

M. Shimbo and K. Hara. 2007. A discriminative learning model for coordinate conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 610–619, Prague, Czech Republic, June.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 195–206, Nancy, France, April.

Exact Inference for Generative Probabilistic Non-Projective Dependency Parsing

Shay B. Cohen
School of Computer Science
Carnegie Mellon University, USA
scohen@cs.cmu.edu

Carlos Gómez-Rodríguez
Departamento de Computación
Universidade da Coruña, Spain
cgomezr@udc.es

Giorgio Satta
Dept. of Information Engineering
University of Padua, Italy
satta@dei.unipd.it

Abstract

We describe a generative model for non-projective dependency parsing based on a simplified version of a transition system that has recently appeared in the literature. We then develop a dynamic programming parsing algorithm for our model, and derive an inside-outside algorithm that can be used for unsupervised learning of non-projective dependency trees.

1 Introduction

Dependency grammars have received considerable attention in the statistical parsing community in recent years. These grammatical formalisms offer a good balance between structural expressivity and processing efficiency. Most notably, when non-projectivity is supported, these formalisms can model crossing syntactic relations that are typical in languages with relatively free word order.

Recent work has reduced non-projective parsing to the identification of a maximum spanning tree in a graph (McDonald et al., 2005; Koo et al., 2007; McDonald and Satta, 2007; Smith and Smith, 2007). An alternative to this approach is to use transition-based parsing (Yamada and Matsumoto, 2003; Nivre and Nilsson, 2005; Attardi, 2006; Nivre, 2009; Gómez-Rodríguez and Nivre, 2010), where there is an incremental processing of a string with a model that scores transitions between parser states, conditioned on the parse history. This paper focuses on the latter approach.

The above work on transition-based parsing has focused on greedy algorithms set in a statistical framework (Nivre, 2008). More recently, dynamic programming has been successfully used for projective parsing (Huang and Sagae, 2010; Kuhlmann et al., 2011). Dynamic programming algorithms for parsing (also known as chart-based algorithms) allow polynomial space representations of all parse trees for a given input string, even in cases where the size of this set is exponential in the length of the string itself. In combination with appropriate semirings, these packed representations can be exploited to compute many values of interest for machine learning, such as best parses and feature expectations (Goodman, 1999; Li and Eisner, 2009).

In this paper we move one step forward with respect to Huang and Sagae (2010) and Kuhlmann et al. (2011) and present a polynomial dynamic programming algorithm for non-projective transition-based parsing. Our algorithm is coupled with a simplified version of the transition system from Attardi (2006), which has high coverage for the type of non-projective structures that appear in various treebanks. Instead of an additional transition operation which permits swapping of two elements in the stack (Titov et al., 2009; Nivre, 2009), Attardi's system allows reduction of elements at non-adjacent positions in the stack. We also present a *generative* probabilistic model for transition-based parsing. The implication for this, for example, is that one can now approach the problem of unsupervised learning of non-projective dependency structures within the transition-based framework.

Dynamic programming algorithms for non-projective parsing have been proposed by Kahane et al. (1998), Gómez-Rodríguez et al. (2009) and Kuhlmann and Satta (2009), but they all run in exponential time in the 'gap degree' of the parsed structures. To the best of our knowledge, this paper is the first to

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1234–1245,
Edinburgh, Scotland, UK, July 27–31, 2011.

introduce a dynamic programming algorithm for inference with non-projective structures of unbounded gap degree.

The rest of this paper is organized as follows. In §2 and §3 we outline the transition-based model we use, together with a probabilistic generative interpretation. In §4 we give the tabular algorithm for parsing, and in §5 we discuss statistical inference using expectation maximization. We then discuss some other aspects of the work in §6 and conclude in §7.

2 Transition-based Dependency Parsing

In this section we briefly introduce the basic definitions for transition-based dependency parsing. For a more detailed presentation of this subject, we refer the reader to Nivre (2008). We then define a specific transition-based model for non-projective dependency parsing that we investigate in this paper.

2.1 General Transition Systems

Assume an input alphabet Σ with a special symbol $\$ \in \Sigma$, which we use as the root of our parse structures. Throughout this paper we denote the input string as $w = a_0 \cdots a_{n-1}$, $n \geq 1$, where $a_0 = \$$ and $a_i \in \Sigma \setminus \{\$\}$ for each i with $1 \leq i \leq n-1$.

A **dependency tree** for w is a directed tree $G_w = (V_w, A_w)$, where $V_w = \{0, \ldots, n-1\}$ is the set of nodes, and $A_w \subseteq V_w \times V_w$ is the set of arcs. The root of G_w is the node 0. The intended meaning is that each node in V_w encodes the position of a token in w. Furthermore, each arc in A_w encodes a dependency relation between two tokens. We write $i \rightarrow j$ to denote a directed arc $(i, j) \in A_w$, where node i is the head and node j is the dependent.

A **transition system** (for dependency parsing) is a tuple $S = (C, T, I, C_t)$, where C is a set of configurations, defined below, T is a finite set of **transitions**, which are partial functions $t\colon C \rightharpoonup C$, I is a total initialization function mapping each input string to a unique initial configuration, and $C_t \subseteq C$ is a set of terminal configurations.

A **configuration** is defined relative to some input string w, and is a triple (σ, β, A), where σ and β are disjoint lists called **stack** and **buffer**, respectively, and $A \subseteq V_w \times V_w$ is a set of arcs. Elements of σ and β are nodes from V_w and, in the case of the stack, a special symbol ¢ that we will use as initial stack symbol. If t is a transition and c_1, c_2 are configurations such that $t(c_1) = c_2$, we write $c_1 \vdash_t c_2$, or simply $c_1 \vdash c_2$ if t is understood from the context.

Given an input string w, a parser based on S incrementally processes w from left to right, starting in the initial configuration $I(w)$. At each step, the parser nondeterministically applies one transition, or else it stops if it has reached some terminal configuration. The dependency graph defined by the arc set associated with a terminal configuration is then returned as one possible analysis for w.

Formally, a **computation** of S is a sequence $\gamma = c_0, \ldots, c_m$, $m \geq 1$, of configurations such that, for every i with $1 \leq i \leq m$, $c_{i-1} \vdash_{t_i} c_i$ for some $t_i \in T$. In other words, each configuration in a computation is obtained as the value of the preceding configuration under some transition. A computation is called **complete** whenever $c_0 = I(w)$ for some input string w, and $c_m \in C_t$.

We can view a transition-based dependency parser as a device mapping strings into graphs (dependency trees). Without any restriction on transition functions in T, these functions might have an infinite domain, and could thus encode even non-recursively enumerable languages. However, in standard practice for natural language parsing, transitions are always specified by some finite mean. In particular, the definition of each transition depends on some finite window at the top of the stack and some finite window at the beginning of the buffer in each configuration. In this case, we can view a transition-based dependency parser as a notational variant of a push-down transducer (Hopcroft et al., 2000), whose computations output sequences that directly encode dependency trees. These transducers are nondeterministic, meaning that several transitions can be applied to some configurations. The transition systems we investigate in this paper follow these principles.

We close this subsection with some additional notation. We denote the stack with its topmost element to the right and the buffer with its first element to the left. We indicate concatenation in the stack and buffer by a vertical bar. For example, for $k \in V_w$, $\sigma|k$ denotes some stack with topmost element k and $k|\beta$ denotes some buffer with first element k. For $0 \leq i \leq n-1$, β_i denotes the buffer

$[i, i+1, \ldots, n-1]$; for $i \geq n$, β_i denotes [] (the empty buffer).

2.2 A Non-projective Transition System

We now turn to give a description of our transition system for non-projective parsing. While a projective dependency tree satisfies the requirement that, for every arc in the tree, there is a directed path between its headword and each of the words between the two endpoints of the arc, a non-projective dependency tree may violate this condition. Even though some natural languages exhibit syntactic phenomena which require non-projective expressive power, most often such a resource is used in a limited way.

This idea is demonstrated by Attardi (2006), who proposes a transition system whose individual transitions can deal with non-projective dependencies only to a limited extent, depending on the distance in the stack of the nodes involved in the newly constructed dependency. The author defines this distance as the **degree** of the transition, with transitions of degree one being able to handle only projective dependencies. This formulation permits parsing a subset of the non-projective trees, where this subset depends on the degree of the transitions. The reported coverage in Attardi (2006) is already very high when the system is restricted to transitions of degree two or three. For instance, on training data for Czech containing 28,934 non-projective relations, 27,181 can be handled by degree two transitions, and 1,668 additional dependencies can be handled by degree three transitions. Table 1 gives additional statistics for treebanks from the CoNLL-X shared task (Buchholz and Marsi, 2006).

We now turn to describe our variant of the transition system of Attardi (2006), which is equivalent to the original system restricted to transitions of degree two. Our results are based on such a restriction. It is not difficult to extend our algorithms (§4) to higher degree transitions, but this comes at the expense of higher complexity. See §6 for more discussion on this issue.

Let $w = a_0 \cdots a_{n-1}$ be an input string over Σ defined as in §2.1, with $a_0 = \$$. Our transition system for non-projective dependency parsing is

$$S^{(\mathrm{np})} = (C, T^{(\mathrm{np})}, I^{(\mathrm{np})}, C_t^{(\mathrm{np})}),$$

Language	Deg. 2	Deg. 3	Deg. 4
Arabic	180	21	7
Bulgarian	961	41	10
Czech	27181	1668	85
Danish	876	136	53
Dutch	9072	2119	171
German	15827	2274	466
Japanese	1484	143	9
Portuguese	3104	424	37
Slovene	601	48	13
Spanish	66	7	0
Swedish	1566	226	79
Turkish	579	185	8

Table 1: The number of non-projective relations of various degrees for several treebanks (training sets), as reported by the parser of Attardi (2006). Deg. stands for 'degree.' The parser did not detect non-projective relations of degree higher than 4.

where C is the same set of configurations defined in §2.1. The initialization function $I^{(\mathrm{np})}$ maps each string w to the initial configuration $([\dot{\varsigma}], \beta_0, \emptyset)$. The set of terminal configurations $C_t^{(\mathrm{np})}$ contains all configurations of the form $([\dot{\varsigma}, 0], [], A)$, for any set of arcs A.

The set of transition functions is defined as

$$T^{(\mathrm{np})} = \{\mathsf{sh}_b \mid b \in \Sigma\} \cup \{\mathsf{la}_1, \mathsf{ra}_1, \mathsf{la}_2, \mathsf{ra}_2\},$$

where each transition is specified below. We let variables i, j, k, l range over V_w, and variable σ is a list of stack elements from $V_w \cup \{\dot{\varsigma}\}$:

$$\begin{aligned}
\mathsf{sh}_b: &\quad (\sigma, k|\beta, A) \vdash (\sigma|k, \beta, A) \text{ if } a_k = b;\\
\mathsf{la}_1: &\quad (\sigma|i|j, \beta, A) \vdash (\sigma|j, \beta, A \cup \{j \rightarrow i\});\\
\mathsf{ra}_1: &\quad (\sigma|i|j, \beta, A) \vdash (\sigma|i, \beta, A \cup \{i \rightarrow j\});\\
\mathsf{la}_2: &\quad (\sigma|i|j|k, \beta, A) \vdash (\sigma|j|k, \beta, A \cup \{k \rightarrow i\});\\
\mathsf{ra}_2: &\quad (\sigma|i|j|k, \beta, A) \vdash (\sigma|i|j, \beta, A \cup \{i \rightarrow k\}).
\end{aligned}$$

Each of the above transitions is undefined on configurations that do not match the forms specified above. As an example, transition la_2 is not defined for a configuration (σ, β, A) with $|\sigma| \leq 2$, and transition sh_b is not defined for a configuration $(\sigma, k|\beta, A)$ with $b \neq a_k$, or for a configuration $(\sigma, [], A)$.

Transition sh_b removes the first node from the buffer, in case this node represents symbol $b \in \Sigma$,

and pushes it into the stack. These transitions are called **shift** transitions. The remaining four transitions are called **reduce** transitions, i.e., transitions that consume nodes from the stack. Notice that in the transition system at hand all the reduce transitions decrease the size of the stack by one element. Transition la_1 creates a new arc with the topmost node on the stack as the head and the second-topmost node as the dependent, and removes the latter from the stack. Transition ra_1 is symmetric with respect to la_1. Transitions la_1 and ra_1 have degree one, as already explained. When restricted to these three transitions, the system is equivalent to the so-called stack-based arc-standard model of Nivre (2004). Transition la_2 and transition ra_2 are very similar to la_1 and ra_1, respectively, but with the difference that they create a new arc between the topmost node in the stack and a node which is two positions below the topmost node. Hence, these transitions have degree two, and are the key components in parsing of non-projective dependencies.

We turn next to describe the equivalence between our system and the system in Attardi (2006). The transition-based parser presented by Attardi pushes back into the buffer elements that are in the top position of the stack. However, a careful analysis shows that only the first position in the buffer can be affected by this operation, in the sense that elements that are pushed back from the stack are never found in buffer positions other than the first. This means that we can consider the first element of the buffer as an additional stack element, always sitting on the top of the top-most stack symbol.

More formally, we can define a function $m_c : C \rightarrow C$ that maps configurations in the original algorithm to those in our variant as follows:

$$m_c((\sigma, k|\beta, A)) = (\sigma|k, \beta, A)$$

By applying this mapping to the source and target configuration of each transition in the original system, it is easy to check that $c_1 \vdash c_2$ in that parser if and only if $m_c(c_1) \vdash m_c(c_2)$ in our variant. We extend this and define an isomorphism between computations in both systems, such that a computation $c_0, \ldots, c_m$ in the original parser is mapped to a computation $m_c(c_0), \ldots, m_c(c_m)$ in the variant, with both generating the same dependency graph A. This proves that our notational variant is in fact equivalent to Attardi's parser.

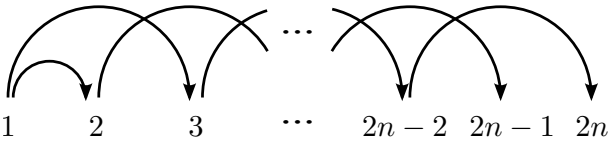


Figure 1: A dependency structure of arbitrary gap degree that can be parsed with Attardi's parser.

A relevant property of the set of dependency structures that can be processed by Attardi's parser, even when restricted to transitions of degree two, is that the number of discontinuities present in each of their subtrees, defined as the **gap degree** by Bodirsky et al. (2005), is not bounded. For example, the dependency graph in Figure 1 has gap degree $n - 1$, and it can be parsed by the algorithm for any arbitrary $n \geq 1$ by applying $2n$ sh_b transitions to push all the nodes into the stack, followed by $(2n - 2)$ ra_2 transitions to create the crossing arcs, and finally one ra_1 transition to create the dependency $1 \rightarrow 2$.

As mentioned in §1, the computational complexity of the dynamic programming algorithm that will be described in later sections does not depend on the gap degree, contrary to the non-projective dependency chart parsers presented by Gómez-Rodríguez et al. (2009) and by Kuhlmann and Satta (2009), whose running time is exponential in the maximum gap degree allowed by the grammar.

3 A Generative Probabilistic Model

In this section we introduce a generative probabilistic model based on the transition system of §2.2. In formal language theory, there is a standard way of giving a probabilistic interpretation to a non-deterministic parser whose computations are based on sequences of elementary operations such as transitions. The idea is to define conditional probability distributions over instances of the transition functions, and to 'combine' these probabilities to assign probabilities to computations and strings.

One difficulty we have to face with when dealing with transition systems is that the notion of computation, defined in §2.1, depends on the input string, because of the buffer component appearing in each configuration. This is a pitfall to generative model-

ing, where we are interested in a system whose computations lead to the *generation* of any string. To overcome this problem, we observe that each computation, defined as a sequence of stacks and buffers (the configurations) can equivalently be expressed as a sequence of stacks and transitions.

More precisely, consider a computation $\gamma = c_0, \ldots, c_m$, $m \geq 1$. Let σ_i, be the stack associated with c_i, for each i with $0 \leq i \leq m$. Let also C_σ be the set of all stacks associated with configurations in C. We can make explicit the transitions that have been used in the computation by rewriting γ in the form $\sigma_0 \vdash_{t_1} \sigma_1 \cdots \sigma_{m-1} \vdash_{t_m} \sigma_m$. In this way, γ generates a string that is composed by all symbols that are pushed into the stack by transitions sh_b, in the left to right order.

We can now associate a probability to (our representation of) sequence γ by setting

$$p(\gamma) = \prod_{i=1}^{m} p(t_i \mid \sigma_{i-1}). \tag{1}$$

To assign probabilities to complete computations we should further multiply $p(\gamma)$ by factors $p_s(\sigma_0)$ and $p_e(\sigma_m)$, where p_s and p_e are start and end probability distributions, respectively, both defined over C_σ. Note however that, as defined in §2.2, all initial configurations are associated with stack $[\dot{\varsigma}]$ and all final configurations are associated with stack $[\dot{\varsigma}, 0]$, thus p_s and p_e are deterministic. Note that the Markov chain represented in Eq. 1 is *homogeneous*, i.e., the probabilities of the transition operations do not depend on the time step.

As a second step we observe that, according to the definition of transition system, each $t \in T$ has an infinite domain. A commonly adopted solution is to introduce a special function, called **history** function and denoted by H, defined over the set C_σ and taking values over some finite set. For each $t \in T$ and $\sigma, \sigma' \in C_\sigma$, we then impose the condition

$$p(t \mid \sigma) = p(t \mid \sigma')$$

whenever $H(\sigma) = H(\sigma')$. Since H is finitely valued, and since T is a finite set, the above condition guarantees that there will only be a finite number of parameters $p(t \mid \sigma)$ in our model.

So far we have presented a general discussion of how to turn a transition-based parser into a generative probabilistic model, and have avoided further specification of the history function. We now turn our attention to the non-projective transition system of §2.2. To actually transform that system into a parametrized probabilistic model, and to develop an associated efficient inference procedure as well, we need to balance between the amount of information we put into the history function and the computational complexity which is required for inference.

We start the discussion with a naïve model using a history function defined by a fixed size window over the topmost portion of the stack. More precisely, each transition is conditioned on the lexical form of the three symbols at the top of the stack σ, indicated as $b_3, b_2, b_1 \in \Sigma$ below, with b_1 referring to the topmost symbol. The parameters of the model are defined as follows.

$$\begin{aligned}
p(\mathsf{sh}_b \mid b_3, b_2, b_1) &= \theta^{\mathsf{sh}_b}_{b_3,b_2,b_1} \ , \quad \forall b \in \Sigma \ , \\
p(\mathsf{la}_1 \mid b_3, b_2, b_1) &= \theta^{\mathsf{la}_1}_{b_3,b_2,b_1} \ , \\
p(\mathsf{ra}_1 \mid b_3, b_2, b_1) &= \theta^{\mathsf{ra}_1}_{b_3,b_2,b_1} \ , \\
p(\mathsf{la}_2 \mid b_3, b_2, b_1) &= \theta^{\mathsf{la}_2}_{b_3,b_2,b_1} \ , \\
p(\mathsf{ra}_2 \mid b_3, b_2, b_1) &= \theta^{\mathsf{ra}_2}_{b_3,b_2,b_1} \ .
\end{aligned}$$

The parameters above are subject to the following normalization conditions, for every choice of $b_3, b_2, b_1 \in \Sigma$:

$$\begin{aligned}
&\theta^{\mathsf{la}_1}_{b_3,b_2,b_1} + \theta^{\mathsf{ra}_1}_{b_3,b_2,b_1} + \theta^{\mathsf{la}_2}_{b_3,b_2,b_1} + \\
&\theta^{\mathsf{ra}_2}_{b_3,b_2,b_1} + \sum_{b \in \Sigma} \theta^{\mathsf{sh}_b}_{b_3,b_2,b_1} = 1 \ .
\end{aligned}$$

This naïve model presents two practical problems. The first problem relates to the efficiency of an inference algorithm, which has a quite high computational complexity, as it will be discussed in §5. A second problem arises in the probabilistic setting. Using this model would require estimating many parameters which are based on trigrams. This leads to higher sample complexity to avoid sparse counts: we would need more samples to accurately estimate the model.

We therefore consider a more elaborated model, which tackles both of the above problems. Again, let $b_3, b_2, b_1 \in \Sigma$ indicate the lexical form of the three symbols at the top of the stack. We define the

distributions $p(t \mid \sigma)$ as follows:

$$p(\mathsf{sh}_b \mid b_1) = \theta_{b_1}^{\mathsf{sh}_b}, \ \ \forall b \in \Sigma \ ,$$
$$p(\mathsf{la}_1 \mid b_2, b_1) = \theta_{b_1}^{\mathsf{rd}} \cdot \theta_{b_2,b_1}^{\mathsf{la}_1} \ ,$$
$$p(\mathsf{ra}_1 \mid b_2, b_1) = \theta_{b_1}^{\mathsf{rd}} \cdot \theta_{b_2,b_1}^{\mathsf{ra}_1} \ ,$$
$$p(\mathsf{la}_2 \mid b_3, b_2, b_1) = \theta_{b_1}^{\mathsf{rd}} \cdot \theta_{b_2,b_1}^{\mathsf{rd}_2} \cdot \theta_{b_3,b_2,b_1}^{\mathsf{la}_2} \ ,$$
$$p(\mathsf{ra}_2 \mid b_3, b_2, b_1) = \theta_{b_1}^{\mathsf{rd}} \cdot \theta_{b_2,b_1}^{\mathsf{rd}_2} \cdot \theta_{b_3,b_2,b_1}^{\mathsf{ra}_2} \ .$$

The parameters above are subject to the following normalization conditions, for every $b_3, b_2, b_1 \in \Sigma$:

$$\sum_{b \in \Sigma} \theta_{b_1}^{\mathsf{sh}_b} + \theta_{b_1}^{\mathsf{rd}} = 1 \ , \tag{2}$$
$$\theta_{b_2,b_1}^{\mathsf{la}_1} + \theta_{b_2,b_1}^{\mathsf{ra}_1} + \theta_{b_2,b_1}^{\mathsf{rd}_2} = 1 \ , \tag{3}$$
$$\theta_{b_3,b_2,b_1}^{\mathsf{la}_2} + \theta_{b_3,b_2,b_1}^{\mathsf{ra}_2} = 1 \ . \tag{4}$$

Intuitively, parameter θ_b^{rd} denotes the probability that we perform a reduce transition instead of a shift transition, given that we have seen lexical form b at the top of the stack. Similarly, parameter $\theta_{b_2,b_1}^{\mathsf{rd}_2}$ denotes the probability that we perform a reduce transition of degree 2 (see §2.2) instead of a reduce transition of degree 1, given that we have seen lexical forms b_1 and b_2 at the top of the stack.

We observe that the above model has a number of parameters $|\Sigma| + 4 \cdot |\Sigma|^2 + 2 \cdot |\Sigma|^3$ (not all independent). This should be contrasted with the naïve model, that has a number of parameters $4 \cdot |\Sigma|^3 + |\Sigma|^4$.

4 Tabular parsing

We present here a dynamic programming algorithm for simulating the computations of the system from §2–3. Given an input string w, our algorithm produces a compact representation of the set $\Gamma(w)$, defined as the set of all possible computations of the model when processing w. In combination with the appropriate semirings, this method can provide for instance the highest probability computation in $\Gamma(w)$, or else the probability of w, defined as the sum of all probabilities of computations in $\Gamma(w)$.

We follow a standard approach in the literature on dynamic programming simulation of stack-based automata (Lang, 1974; Tomita, 1986; Billot and Lang, 1989). More recently, this approach has also been applied by Huang and Sagae (2010) and by Kuhlmann et al. (2011) to the simulation of projective transition-based parsers. The basic idea in this approach is to decompose computations of the parser into smaller parts, group them into equivalence classes and recombine to obtain larger parts of computations.

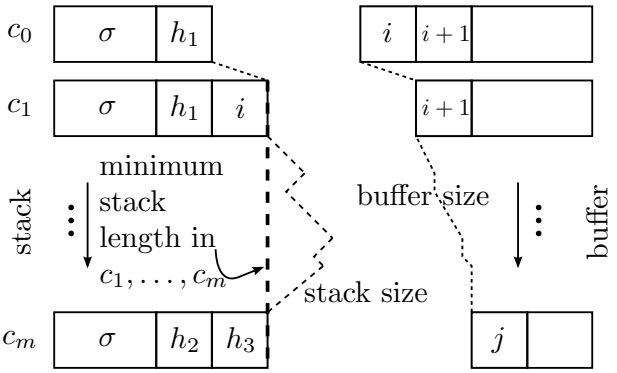


Figure 2: Schematic representation of the computations γ associated with item $[h_1, i, h_2 h_3, j]$.

Let $w = a_0 \cdots a_{n-1}$, V_w and $S^{(np)}$ be defined as in §2. We use a structure called **item**, defined as

$$[h_1, i, h_2 h_3, j],$$

where $0 \leq i < j \leq n$ and $h_1, h_2, h_3 \in V_w$ must satisfy $h_1 < i$ and $i \leq h_2 < h_3 < j$. The intended interpretation of an item can be stated as follows; see also Figure 2.

- There exists a computation γ of $S^{(\mathrm{np})}$ on w having the form $c_0, \ldots, c_m$, $m \geq 1$, with $c_0 = (\sigma | h_1, \beta_i, A)$ and $c_m = (\sigma | h_2 | h_3, \beta_j, A')$ for some stack σ and some arc sets A and A';
- For each i with $1 \leq i < m$, the stack σ_i associated with configuration c_i has the list σ at the bottom and satisfies $|\sigma_i| \geq |\sigma| + 2$.

Some comments on the above conditions are in order here. Let $t_1, \cdots, t_m$ be the sequence of transitions in $T^{(np)}$ associated with computation γ. Then we have $t_1 = \mathsf{sh}_{a_i}$, since $|\sigma_1| \geq |\sigma| + 2$. Thus we conclude that $|\sigma_1| = |\sigma| + 2$.

The most important consequence of the definition of item is that each transition t_i with $2 \leq i \leq m$ does not depend on the content of the σ portion of the stack σ_i. To see this, consider transition $c_{i-1} \vdash_{t_i} c_i$. If $t_i = \mathsf{sh}_{a_i}$, the content of σ is irrelevant at

this step, since in our model sh_{a_i} is conditioned only on the topmost stack symbol of σ_{i-1}, and we have $|\sigma_{i-1}| \geq |\sigma| + 2$.

Consider now the case of $t_i = \mathsf{la}_2$. From $|\sigma_i| \geq |\sigma| + 2$ we have that $|\sigma_{i-1}| \geq |\sigma| + 3$. Again, the content of σ is irrelevant at this step, since in our model la_2 is conditioned only on the three topmost stack symbols of σ_{i-1}. A similar argument applies to the cases of $t_i \in \{\mathsf{ra}_2, \mathsf{la}_1, \mathsf{ra}_1\}$.

From the above, we conclude that if we apply the transitions $t_1, \ldots, t_m$ to stacks of the form $\sigma|h_1$, the resulting computations have all identical probabilities, independently of the choice of σ.

Each computation satisfying the two conditions above will be called an **I-computation** associated with item $[h_1, i, h_2 h_3, j]$. Notice that an I-computation has the overall effect of replacing node h_1 sitting above a stack σ with nodes h_2 and h_3. This is the key property in the development of our algorithm below.

We specify our dynamic programming algorithm as a deduction system (Shieber et al., 1995). The deduction system starts with axiom $[\dot{\varsigma}, 0, \dot{\varsigma}0, 1]$, corresponding to an initial stack $[\dot{\varsigma}]$ and to the shift of $a_0 = \$$ from the buffer into the stack. The set $\Gamma(w)$ is non-empty if and only if item $[\dot{\varsigma}, 0, \dot{\varsigma}0, n]$ can be derived using the inference rules specified below. Each inference rule is annotated with the type of transition it simulates, along with the arc constructed by the transition itself, if any.

$$\frac{[h_1, i, h_2 h_3, j]}{[h_3, j, h_3 j, j+1]} \ (\mathsf{sh}_{a_j})$$

$$\frac{[h_1, i, h_2 h_3, k] \quad [h_3, k, h_4 h_5, j]}{[h_1, i, h_2 h_5, j]} \ (\mathsf{la}_1; h_5 \rightarrow h_4)$$

$$\frac{[h_1, i, h_2 h_3, k] \quad [h_3, k, h_4 h_5, j]}{[h_1, i, h_2 h_4, j]} \ (\mathsf{ra}_1; h_4 \rightarrow h_5)$$

$$\frac{[h_1, i, h_2 h_3, k] \quad [h_3, k, h_4 h_5, j]}{[h_1, i, h_4 h_5, j]} \ (\mathsf{la}_2; h_5 \rightarrow h_2)$$

$$\frac{[h_1, i, h_2 h_3, k] \quad [h_3, k, h_4 h_5, j]}{[h_1, i, h_2 h_4, j]} \ (\mathsf{ra}_2; h_2 \rightarrow h_5)$$

The above deduction system infers items in a bottom-up fashion. This means that longer computations over substrings of w are built by combining shorter ones. In particular, the inference rule sh_{a_j} asserts the existence of I-computations consisting of a single sh_{a_j} transition. Such computations are represented by the consequent item $[h_3, j, h_3 j, j+1]$, indicating that the index of the shifted word a_j is added to the stack by pushing it on top of h_3.

The remaining four rules implement the reduce transitions of the model. We have already observed in §2.2 that all available reduce transitions shorten the size of the stack by one unit. This allows us to combine pairs of I-computations with a reduce transition, resulting in a computation that is again an I-computation. More precisely, if we concatenate an I-computation asserted by an item $[h_1, i, h_2 h_3, k]$ with an I-computation asserted by an item $[h_3, k, h_4 h_5, j]$, we obtain a computation that has the overall effect of increasing the size of the stack by 2, replacing the topmost stack element h_1 with stack elements h_2, h_4 and h_5. If we now apply any of the reduce transitions from the inventory of the model, we will remove one of these three nodes from the stack, and the overall result will be again an I-computation, which can then be asserted by a certain item. For example, if we apply the reduce transition la_1, the consequent item is $[h_1, i, h_2 h_5, j]$, since an la_1 transition removes the second topmost element from the stack (h_4). The other reduce transitions remove a different element, and thus their rules produce different consequent items.

The above argument shows the soundness of the deduction system, i.e., an item $I = [h_1, i, h_2 h_3, j]$ is only generated if there exists an I-computation $\gamma = c_0, \ldots, c_m$ with $c_0 = (\sigma|h_1, \beta_i, A)$ and $c_m = (\sigma|h_2|h_3, \beta_j, A')$. To prove completeness, we must show the converse result, i.e., that the existence of an I-computation γ implies that item I is inferred. We first do this under the assumption that the inference rule for the shift transitions do not have an antecedent, i.e., items $[h_1, j, h_1 j, j+1]$ are considered as axioms. We proceed by using strong induction on the length m of the computation γ.

For $m = 1$, γ consists of a single transition sh_{a_j}, and the corresponding item $I = [h_1, j, h_1 j, j+1]$ is constructed as an axiom. For $m > 1$, let γ be as specified above. The transition that produced

c_m must have been a reduce transition, otherwise γ would not be an I-computation. Let c_k be the rightmost configuration in $c_0, \ldots, c_{m-1}$ whose stack size is $|\sigma| + 2$. Then it can be shown that the computations $\gamma_1 = c_0, \ldots, c_k$ and $\gamma_2 = c_k, \ldots, c_{m-1}$ are again I-computations. Since γ_1 and γ_2 have strictly fewer transitions than γ, by the induction hypothesis, the system constructs items $[h_1, i, h_2 h_3, k]$ and $[h_3, k, h_4 h_5, j]$, where h_2 and h_3 are the stack elements at the top of c_k. Applying to these items the inference rule corresponding to the reduce transition at hand, we can construct item I.

When the inference rule for the shift transition has an antecedent $[h_1, i, h_2 h_3, j]$, as indicated above, we have the overall effect that I-computations consisting of a single transition shifting a_j on the top of h_3 are simulated only in case there exists a computation starting with configuration $([\text{¢}], \beta_0)$ and reaching a configuration of the form $(\sigma | h_2 | h_3, \beta_j)$. This acts as a filter on the search space of the algorithm, but does not invalidate the completeness property. However, in this case the proof is considerably more involved, and we do not report it here.

An important property of the deduction system above, which will be used in the next section, is that the system is unambiguous, that is, each **I-computation** is constructed by the system in a unique way. This can be seen by observing that, in the sketch of the completeness proof reported above, there always is an unique choice of c_k that decomposes I-computation γ into I-computations γ_1 and γ_2. In fact, if we choose a configuration $c_{k'}$ other than c_k with stack size $|\sigma| + 2$, the computation $\gamma_2' = c_{k'}, \ldots, c_{m-1}$ will contain c_k as an intermediate configuration, which violates the definition of I-computation because of an intervening stack having size not larger than the size of the stack associated with the initial configuration.

As a final remark, we observe that we can keep track of all inference rules that have been applied in the computation of each item by the above algorithm, by encoding each application of a rule as a reference to the pair of items that were taken as antecedent in the inference. In this way, we obtain a parse forest structure that can be viewed as a hypergraph or as a non-recursive context-free grammar, similar to the case of parsing based on context-free grammars. See for instance Klein and Manning (2001) or Nederhof (2003). Such a parse forest encodes all valid computations in $\Gamma(w)$, as desired.

The algorithm runs in $\mathcal{O}(n^8)$ time. Using methods similar to those specified in Eisner and Satta (1999), we can reduce the running time to $\mathcal{O}(n^7)$. However, we do not further pursue this idea here, and proceed with the discussion of exact inference, found in the next section.

5 Inference

We turn next to specify exact inference with our model, for computing feature expectations. Such inference enables, for example, the derivation of an expectation-maximization algorithm for unsupervised parsing.

Here, a feature is a function over computations, providing the count of a pattern related to a parameter. We denote by $f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma)$, for instance, the number of occurrences of transition la_2 within γ with topmost stack symbols having word forms $b_3, b_2, b_1 \in \Sigma$, with b_1 associated with the topmost stack symbol.

Feature expectations are computed by using an inside-outside algorithm for the items in the tabular algorithm. More specifically, given a string w, we associate each item $[h_1, i, h_2 h_3, j]$ defined as in §4 with two quantities:

$$I([h_1, i, h_2 h_3, j]) = \sum_{\gamma = ([h_1], \beta_i), \ldots, ([h_2, h_3], \beta_j)} p(\gamma) \; ; \tag{5}$$

$$O([h_1, i, h_2 h_3, j]) = \sum_{\substack{\sigma, \gamma = ([\text{¢}], \beta_0), \ldots, (\sigma | h_1, \beta_i) \\ \gamma' = (\sigma | h_2 | h_3, \beta_j), \ldots, ([\text{¢}, 0], \beta_n)}} p(\gamma) \cdot p(\gamma') \; . \tag{6}$$

$I([h_1, i, h_2 h_3, j])$ and $O([h_1, i, h_2 h_3, j])$ are called the **inside** and the **outside** probabilities, respectively, of item $[h_1, i, h_2 h_3, j]$. The tabular algorithm of §4 can be used to compute the inside probabilities. Using the gradient transformation (Eisner et al., 2005), a technique for deriving outside probabilities from a set of inference rules, we can also compute $O([h_1, i, h_2 h_3, j])$. The use of the gradient transformation is valid in our case because the tabular algorithm is unambiguous (see §4).

Using the inside and outside probabilities, we can now efficiently compute feature expectations for our

$$E_{p(\gamma|w)}[f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma)] = \sum_{\gamma \in \Gamma(w)} p(\gamma \mid w) \cdot f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma) = \frac{1}{p(w)} \cdot \sum_{\gamma \in \Gamma(w)} p(\gamma) \cdot f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma)$$

$$= \frac{1}{p(w)} \cdot \sum_{\substack{\sigma,i,k,j,\\ h_1,h_2,h_3,h_4,h_5,\\ \text{s.t. } a_{h_2}=b_3,\\ a_{h_4}=b_2,\, a_{h_5}=b_1}} \sum_{\substack{\gamma_0=([\c{c}],\beta_0),\ldots,(\sigma|h_1,\beta_i),\\ \gamma_1=(\sigma|h_1,\beta_i),\ldots,(\sigma|h_2|h_3,\beta_k),\\ \gamma_2=(\sigma|h_2|h_3,\beta_k),\ldots,(\sigma|h_2|h_4|h_5,\beta_j),\\ \gamma_3=(\sigma|h_2|h_5,\beta_j),\ldots,([\c{c},0],\beta_n)}} p(\gamma_0) \cdot p(\gamma_1) \cdot p(\gamma_2) \cdot p(\mathsf{la}_2 \mid b_3, b_2, b_1) \cdot p(\gamma_3)$$

$$= \frac{\theta^{\mathsf{rd}}_{b_1} \cdot \theta^{\mathsf{rd}_2}_{b_2,b_1} \cdot \theta^{\mathsf{la}_2}_{b_3,b_2,b_1}}{p(w)} \cdot \sum_{\substack{\sigma,i,j,\\ h_1,h_2,h_5,\text{ s.t.}\\ a_{h_2}=b_3,\, a_{h_5}=b_1}} \sum_{\substack{\gamma_0=([\c{c}],\beta_0),\ldots,(\sigma|h_1,\beta_i),\\ \gamma_3=(\sigma|h_2|h_5,\beta_j),\ldots,([\c{c},0],\beta_n)}} p(\gamma_0) \cdot p(\gamma_3) \cdot$$

$$\cdot \sum_{\substack{k,h_3,h_4,\\ \text{s.t. } a_{h_4}=b_2}} \sum_{\gamma_1=(\sigma|h_1,\beta_i),\ldots,(\sigma|h_2|h_3,\beta_k)} p(\gamma_1) \cdot \sum_{\gamma_2=(\sigma|h_2|h_3,\beta_k),\ldots,(\sigma|h_2|h_4|h_5,\beta_j)} p(\gamma_2)$$

Figure 3: Decomposition of the feature expectation $E_{p(\gamma|w)}[f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma)]$ into a finite summation. Quantity $p(w)$ above is the sum over all probabilities of computations in $\Gamma(w)$.

model. Figure 3 shows how to express the expectation of feature $f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma)$ by means of a finite summation. Using Eq. 5 and 6 and the relation $p(w) = I([\c{c}, 0, \c{c}0, n])$ we can then write:

$$E_{p(\gamma|w)}[f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma)] = \frac{\theta^{\mathsf{rd}}_{b_1} \cdot \theta^{\mathsf{rd}_2}_{b_2,b_1} \cdot \theta^{\mathsf{la}_2}_{b_3,b_2,b_1}}{I([\c{c}, 0, \c{c}0, n])} \cdot$$

$$\cdot \sum_{\substack{i,j,h_1,h_4,h_5,\\ \text{s.t. } a_{h_4}=b_2,\, a_{h_5}=b_1}} O([h_1, i, h_4 h_5, j]) \cdot$$

$$\cdot \sum_{\substack{k,h_2,h_3,\\ \text{s.t. } a_{h_2}=b_3}} I([h_1, i, h_2 h_3, k]) \cdot I([h_3, k, h_4 h_5, j]) \; .$$

Very similar expressions can be derived for the expectations for features $f^{\mathsf{ra}_2}_{b_3,b_2,b_1}(\gamma)$, $f^{\mathsf{la}_1}_{b_2,b_1}(\gamma)$, and $f^{\mathsf{ra}_1}_{b_2,b_1}(\gamma)$. As for feature $f^{\mathsf{sh}_b}_{b_1}(\gamma)$, $b \in \Sigma$, the above approach leads to

$$E_{p(\gamma|w)}[f^{\mathsf{sh}_b}_{b_1}(\gamma)] =$$

$$= \frac{\theta^{\mathsf{sh}_b}_{b_1}}{I([\c{c}, 0, \c{c}0, n])} \cdot \sum_{\substack{\sigma,i,h,\text{ s.t.}\\ a_h=b_1,\, a_i=b}} O([h, i, hi, i+1]) \; .$$

As mentioned above, these expectations can be used, for example, to derive an EM algorithm for our model. The EM algorithm in our case is not completely straightforward because of the way we parametrize the model. We give now the re-estimation steps for such an EM algorithm. We assume that all expectations below are taken with respect to a set of parameters $\boldsymbol{\theta}$ from iteration $s-1$ of the algorithm, and we are required to update these $\boldsymbol{\theta}$. To simplify notation, let us assume that there is only one string w in the training corpus. For each $b_1 \in \Sigma$, we define:

$$Z_{b_1} = \sum_{b_2 \in \Sigma} E_{p(\gamma|w)} \left[f^{\mathsf{la}_1}_{b_2,b_1}(\gamma) + f^{\mathsf{ra}_1}_{b_2,b_1}(\gamma) \right]$$

$$+ \sum_{b_3,b_2 \in \Sigma} E_{p(\gamma|w)} \left[f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma) + f^{\mathsf{ra}_2}_{b_3,b_2,b_1}(\gamma) \right] \; ;$$

$$Z_{b_2,b_1} = \sum_{b_3 \in \Sigma} E_{p(\gamma|w)} \left[f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma) + f^{\mathsf{ra}_2}_{b_3,b_2,b_1}(\gamma) \right] \; .$$

We then have, for every $b \in \Sigma$:

$$\theta^{\mathsf{sh}_b}_{b_1}(s) \leftarrow \frac{E_{p(\gamma|w)}[f^{\mathsf{sh}_b}_{b_1}(\gamma)]}{Z_{b_1} + \sum_{b' \in \Sigma} E_{p(\gamma|w)}[f^{\mathsf{sh}_{b'}}_{b_1}(\gamma)]} \; .$$

Furthermore, we have:

$$\theta^{\mathsf{la}_1}_{b_2,b_1}(s) \leftarrow \frac{E_{p(\gamma|w)}[f^{\mathsf{la}_1}_{b_2,b_1}(\gamma)]}{Z_{b_2,b_1} + E_{p(\gamma|w)}\left[f^{\mathsf{la}_1}_{b_2,b_1}(\gamma) + f^{\mathsf{ra}_1}_{b_2,b_1}(\gamma)\right]},$$

and:

$$\theta^{\mathsf{la}_2}_{b_3,b_2,b_1}(s) \leftarrow \frac{E_{p(\gamma|w)}[f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma)]}{E_{p(\gamma|w)}\left[f^{\mathsf{la}_2}_{b_3,b_2,b_1}(\gamma) + f^{\mathsf{ra}_2}_{b_3,b_2,b_1}(\gamma)\right]}.$$

The rest of the parameter updates can easily be derived using the above updates because of the sum-to-1 constraints in Eq. 2–4.

6 Discussion

We note that our model inherits spurious ambiguity from Attardi's model. More specifically, we can have different derivations, corresponding to different system computations, that result in identical dependency graphs and strings. While running our tabular algorithm with the Viterbi semiring efficiently computes the highest probability computation in $\Gamma(w)$, spurious ambiguity means that finding the highest probability dependency tree is NP-hard. This latter result can be shown using proof techniques similar to those developed by Sima'an (1996). We leave it for future work how to eliminate spurious ambiguity from the model.

While in the previous sections we have described a tabular method for the transition system of Attardi (2006) restricted to transitions of degree up to two, it is possible to generalize the model to include higher-degree transitions. In the general formulation of Attardi parser, transitions of degree d create links involving nodes located d positions beneath the topmost position in the stack:

$$\mathsf{la}_d: \quad (\sigma|i_1|i_2|\ldots|i_{d+1},\beta,A) \vdash (\sigma|i_2|\ldots|i_{d+1},\beta,A\cup\{i_{d+1}\rightarrow i_1\});$$

$$\mathsf{ra}_d: \quad (\sigma|i_1|i_2|\ldots|i_{d+1},\beta,A) \vdash (\sigma|i_1|i_2|\ldots|i_d,\beta,A\cup\{i_1\rightarrow i_{d+1}\}).$$

To define a transition system that supports transitions up to degree D, we use a set of items of the form $[s_1\ldots s_{D-1}, i, e_1\ldots e_D, j]$, corresponding (in the sense of §4) to computations of the form $c_0,\ldots,c_m$, $m \geq 1$, with $c_0 = (\sigma|s_1|\ldots|s_{D-1},\beta_i,A)$ and $c_m = (\sigma|e_1|\ldots|e_D,\beta_j,A')$. The deduction steps corresponding to reduce transitions in this general system have the general form

$$\frac{[s_1\ldots s_{D-1}, i, e_1 m_1\ldots m_{D-1}, j] \quad [m_1\ldots m_{D-1}, j, e_2\ldots e_{D+1}, w]}{[s_1\ldots s_{D-1}, i, e_1\ldots e_{c-1}e_{c+1}\ldots e_{D+1}, w]}\ (e_p\rightarrow e_c)$$

where the values of p and c differ for each transition: to obtain the inference rule corresponding to a la_d transition, we make $p = D+1$ and $c = D+1-d$; and to obtain the rule for a ra_d transition, we make $p = D+1-d$ and $c = D+1$. Note that the parser runs in time $O(n^{3D+2})$, where D stands for the maximum transition degree, so each unit increase in the transition degree adds a cubic factor to the parser's polynomial time complexity. This is in contrast to a previous tabular formulation of the Attardi parser by Gómez-Rodríguez et al. (2011), which ran in exponential time.

The model for the transition system we give in this paper is generative. It is not hard to naturally extend this model to the discriminative setting. In this case, we would condition the model on the input string to get a conditional distribution over derivations. It is perhaps more natural in this setting to use arbitrary weights for the parameter values, since the computation of a normalization constant (the probability of a string) is required in any case. Arbitrary weights in the generative setting could be more problematic, because it would require computing a normalization constant corresponding to a sum over all *strings* and derivations.

7 Conclusion

We presented in this paper a generative probabilistic model for non-projective parsing, together with the description of an efficient tabular algorithm for parsing and doing statistical inference with the model.

Acknowledgments

The authors thank Marco Kuhlmann for helpful comments on an early draft of the paper. The authors also thank Giuseppe Attardi for the help received to extract the parsing statistics. The second author has been partially supported by Ministerio de Ciencia e Innovación and FEDER (TIN2010-18552-C03-02).

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170, New York, USA.

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 143–151, Vancouver, Canada.

Manuel Bodirsky, Marco Kuhlmann, and Mathias Möhl. 2005. Well-nested drawings as models of syntactic structure. In *Tenth Conference on Formal Grammar and Ninth Meeting on Mathematics of Language*, pages 195–203, Edinburgh, UK.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164, New York, USA.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and Head Automaton Grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, College Park, MD, USA.

Jason Eisner, Eric Goldlust, and Noah A. Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proceedings of HLT-EMNLP*, pages 281–290.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1492–1501, Uppsala, Sweden.

Carlos Gómez-Rodríguez, David J. Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 291–299, Athens, Greece.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics* (in press), 37(3).

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2000. *Introduction to Automata Theory*. Addison-Wesley, 2nd edition.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1077–1086, Uppsala, Sweden.

Sylvain Kahane, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *36th Annual Meeting of the Association for Computational Linguistics and 18th International Conference on Computational Linguistics (COLING-ACL)*, pages 646–652, Montréal, Canada.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of the IWPT*, pages 123–134.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the EMNLP-CoNLL*, pages 141–150.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 478–486, Athens, Greece.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, Portland, Oregon, USA.

Bernard Lang. 1974. Deterministic techniques for efficient non-deterministic parsers. In Jacques Loecx, editor, *Automata, Languages and Programming, 2nd Colloquium, University of Saarbrücken, July 29–August 2, 1974*, number 14 in Lecture Notes in Computer Science, pages 255–269. Springer.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 40–51, Singapore.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Tenth International Conference on Parsing Technologies (IWPT)*, pages 121–132, Prague, Czech Republic.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Human Language Technology Conference (HLT) and Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–530, Vancouver, Canada.

Mark-Jan Nederhof. 2003. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29(1):135–143.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106, Ann Arbor, USA.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the 47th Annual Meeting of the ACL and the Fourth International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Singapore.

Stuart M. Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36.

Khalil Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proceedings of COLING*, pages 1175–1180.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proceedings of the EMNLP-CoNLL*, pages 132–140.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of IJCAI*, pages 281–290.

Masaru Tomita. 1986. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*. Springer.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT)*, pages 195–206, Nancy, France.

Semi-supervised CCG Lexicon Extension

Emily Thomforde
University of Edinburgh
e.j.thomforde@sms.ed.ac.uk

Mark Steedman
University of Edinburgh
steedman@inf.ed.ac.uk

Abstract

This paper introduces Chart Inference (CI), an algorithm for deriving a CCG category for an unknown word from a partial parse chart. It is shown to be faster and more precise than a baseline brute-force method, and to achieve wider coverage than a rule-based system. In addition, we show the application of CI to a domain adaptation task for question words, which are largely missing in the Penn Treebank. When used in combination with self-training, CI increases the precision of the baseline StatCCG parser over subject-extraction questions by 50%. An error analysis shows that CI contributes to the increase by expanding the number of category types available to the parser, while self-training adjusts the counts.

1 Introduction

Unseen lexical items are a major cause of error in strongly lexicalised parsers such as those based on CCG (Clark and Curran, 2003; Hockenmaier, 2003). The problem is especially acute for less privileged languages, but even in the case of English, we are aware of many category types entirely missing from the Penn Treebank (Clark et al., 2004).

In the case of totally unseen words, the standard method used by StatCCG (Hockenmaier, 2003) and many other treebank parsers is part-of-speech backoff, which is quite effective, affording an F-score of 93% over dependencies in §00 in the optimal configuration. It is difficult to say how backing off affects dependency errors, but when we examine category match accuracy of the CCGBank-trained parser, we find that POS backoff has been used on 19.6% of tokens, which means that those tokens are unseen, or too infrequent in the training data to be included in the lexicon. Of the 3320 items the parser labelled incorrectly, 675 (20.3%) are words that are missing from the lexicon entirely.[1] In the best case, if we were able to learn lexical entries for those 675, we could transfer them to lexical treatment, which is 93.5% accurate, rather than POS backoff, which is 89.3% accurate. Under these conditions, we predict a further 631 word/category pairs to be tagged correctly by the parser, reducing the error rate from 7.4% to 6% on §00. Further to reducing parsing error, a robust method for learning words from unlabelled data would result in the recovery of interesting and important category types that are missing from our standard lexical resources.

This paper introduces Chart Inference (CI) as a strategy for deducing a ranked set of possible categories for an unknown word using the partial chart formed from the known words that surround it. CCG (Steedman, 2000) is particularly suited to this problem, because category types can be inferred from the types of the surrounding constituents. CI is designed to take advantage of this property of generative CCGBank-trained parser, and of access to the full inventory of CCG combinators and non-combinatory unary rules from the trained model. It is capable of learning category types that are completely missing from the lexicon, and is superior to existing learning systems in both precision and efficiency.

Four experiments are discussed in this paper. The first compares three word-learning methods for their ability to converge to a toy target lexicon. The sec-

[1] A further 269 (8%) are cases where the word is known, but has not been seen with the correct category.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1246–1256,
Edinburgh, Scotland, UK, July 27–31, 2011.

ond and third compare the three methods based on their ability to correctly tag the all the words in a small natural language corpus. The final experiment shows how Chart Induction can be effectively used in a domain adaptation task where a small number of category types are known to be missing from the lexicon.

2 Learning Words

The methods used in this paper all operate under a restricted learning setting, over sentences where all but one word is in the lexicon. Since the learning portion of the algorithm is unsupervised, it has access to an essentially unlimited amount of unlabelled data, and it can afford to skip any sentence that does not conform to the one-unseen-word restriction. Attempting two or more OOL words at a time from one sentence would compound the search space and the error rate. We do not address the much harder problem of hypothesising missing categories for known words, which should presumably be handled by quite other methods, such as prior offline generalization of the lexicon.

2.1 A Brute-force System

One of the early lexical acquisition systems using Categorial Grammar was that of Watkinson and Manandhar (1999; 2000; 2001a; 2001b). This system attempted to simultaneously learn a CG lexicon and annotate unlabelled text with parse derivations. Using a stripped-down parser that only utilised the forward- and backward-application rules, they iteratively learned the lexicon from the feedback from online parsing. The system decided which parse was best based on the lexicon, and then decided which additions to the lexicon to make based on principles of compression. After each change, the system reexamined the parses for previous sentences and updated them to reflect the new lexicon.

They report fully convergent results on two toy corpora, but the parsing accuracy of the system trained on natural language data was far below the state of the art. However, they do show categorial grammar to be a promising basis for artificial language acquisition, because CCG makes learning the lexicon and learning the grammar the same task (Watkinson and Manandhar, 1999). They also showed that seeding the lexicon with examples of lexical items (closed-class words in their case), rather than just a list of possible category types, increased its chances of converging. This approach of automating the learning process differs from the previous language learning methods described, in that it doesn't require the specification of any particular patterns, only knowledge of the grammar formalism.

For this paper, as a baseline, we implement a generalised version of Watkinson and Manandhar's mechanism for determining the category γ of a single OOL word in a sentence where the rest of the words $C_1 ... C_N$ are in the lexicon: $\gamma = arg\max Parse(C_1 ... C_n, \gamma)$. This is equivalent to backing off to the set of all known category types; the learner returns the category that maximises the probability of the completed parse tree. We ignore the optimisation and compression steps of the original system.

2.2 A Rule-based System

Yao et al. (2009a; 2009b) developed a learning system based on handwritten translation rules for deducing the category (X) of a single unknown word in a sentence consisting of a sequence of partially-parsed constituents (A..N).

Their system was based on a small inventory of inference rules that eliminated ambiguity in the ordering of arguments. For example, one of the Level 3 inference rules specifies the order of the arguments in the deduced category:

$A\ \mathbf{X}\ B\ C \rightarrow D \Rightarrow \mathbf{X} = ((D\backslash A)/C)/B$

Without this inductive bias the learner would have to deal with the ambiguity of the options $((D/C)/B)\backslash A$ and $((D/C)\backslash A)/B$ at minimum. In addition they limited their learner to CG-compatible parse structures and their constituent strings to length 4.

Their argument is that only this minimal bias is needed to learn syntactic structures, including the fronting of polar interrogative auxiliaries and auxiliary word order (should $>$ have $>$ been), from a training set that did not explicitly contain full evidence for them.

Although Yao et al. (2009b) used the full set of CCG combinators to generate learned categories, they employed a post-processing step to filter spurious categories by checking whether the category

$\text{DERIVE}([C_1 ... C_n], \beta, \gamma)$

if $\beta = \emptyset$
then return (γ)
else if $C_1 = C_n = X$
then $\begin{cases} \gamma = \gamma + \beta; \\ \text{DERIVE}(X, \emptyset, \gamma) \end{cases}$
else $\begin{cases} \textbf{if } C_1 \notin S, X \\ \quad \textbf{then } \text{DERIVE}([C_2 ... C_n], \beta \backslash C_1, \gamma) \\ \textbf{if } C_n \notin S, X \\ \quad \textbf{then } \text{DERIVE}([C_1 ... C_{n-1}], \beta / C_n, \gamma) \\ \textbf{if } \beta \equiv B \textbf{ and } C_1 \equiv B/A \textbf{ and } C_1 \notin S, X \\ \quad \textbf{then } \text{DERIVE}([C_2 ... C_n], A, \gamma) \\ \textbf{if } \beta \equiv B \textbf{ and } C_n \equiv B \backslash A \textbf{ and } C_n \notin S, X \\ \quad \textbf{then } \text{DERIVE}([C_1 ... C_{n-1}], A, \gamma) \end{cases}$

Figure 1: Generalised recursive rule-based algorithm, where $[C_1 ... C_n]$ is a sequence of categories, one of which is X, β is a result category, and γ is the (initially empty) category set.

participated in a CG-only derivation (using application rules only). This is effective in limiting spurious derivations, but at the expense of reduced recall on those sentences for whose analysis CCG rules of composition etc. are crucial.

Their rules were effective for their toy-scale datasets, but for the purposes of this paper we have implemented a generalised version of the recursive algorithm for use in wide-coverage parsing. This algorithm is outlined in Figure 1. It takes a sequence of categorial constituents, all known except one (**X**), and builds a candidate set of categories (γ) for the unknown word by recursively applying Yao's Level 0 and Level 1 inference rules.

2.3 Chart Inference

Both Watkinson's and Yao's experiments were fully convergent over toy datasets, but did not scale to realistic corpora. Watkinson attempted to learn from the LLL corpus (Kazakov et al., 1998), but attributed the failure to the small amount of training data relative to the corpus, and the naive initial category set. Yao's method was only ever designed as a proof-of-concept to show how much of the language can be learned from partial evidence, and was not meant to be run in earnest in a real-world learning setting. For one, the rules do not cover the full set of partial parse conditions, and further to that, they do not allow for partial parses to be reanalysed within the learning framework.

To that end, we have developed a learning algorithm that is capable of operating within the one-unknown-word-per-sentence learning setting established by the two baseline systems, that is able to invent new category types, and that is able to take advantage of the full generality of CCG. This section shows that it performs as well as the previous two systems on a toy corpus, and the next section proves that it more readily scales to natural language domains.

Mellish (1989) established a two-stage bidirectional chart parser for diagnosing errors in input text. His method relied heavily on heuristic rules, and the only evaluation he did was on number of cycles needed for each type of error, and number of solutions produced. His method was designed for use in producing parses where the original parser failed, dealing with omissions, insertions, and misspelled/unknown words. The only method used to rank the possible solutions was heuristic scores.

Kato (1994) implemented a revised system that used a generalised top-down parser, rather than a chart, and was able to get the number of cycles to decrease.

In both cases the evaluation was only on a toy corpus, and they did not evaluate on whether the systems diagnosed the errors correctly, or whether the solution they offered was accurate. They also had to deal with cases where the error was ambiguous, for example, where an inserted word could be interpreted as a misspelling or vice-versa.

Where Mellish uses the two-stage parsing process to complete malformed parses, we use it to diagnose unknown lexical items. In addition, we work on the scale of a full grammar and wide-coverage parser, using modern lexical corpora.

Our method is a wrapper for a naive generative CCG parser StatOpenCCG (Christodoulopoulos, 2008), a statistical extension to OpenCCG (White and Baldridge, 2003). In the general case, the parser is trained on all the labelled data available in a particular learning setting, then the learner discovers new lexical items from unlabelled text. Like the brute force and rule-based systems, it is vulnerable

CCG Combinator					**Inverse Combinator**						
A/B	B	$\rightarrow$	A	(>)	$\mathbf{X}$	B	$\rightarrow$	A	$\Rightarrow$	$\mathbf{X}=A/B$	if v(B) ≤ 1
					A/B	$\mathbf{X}$	$\rightarrow$	A	$\Rightarrow$	$\mathbf{X}=B$	
B	$A\backslash B$	$\rightarrow$	A	(<)	$\mathbf{X}$	$A\backslash B$	$\rightarrow$	A	$\Rightarrow$	$\mathbf{X}=B$	
					B	$\mathbf{X}$	$\rightarrow$	A	$\Rightarrow$	$\mathbf{X}=A\backslash B$	if v(B) ≤ 1
A/C	C/B	$\rightarrow$	A/B	(>B)	$\mathbf{X}$	C/B	$\rightarrow$	A/B	$\Rightarrow$	$\mathbf{X}=A/C$	
					A/C	$\mathbf{X}$	$\rightarrow$	A/B	$\Rightarrow$	$\mathbf{X}=C/B$	
$C\backslash B$	$A\backslash C$	$\rightarrow$	$A\backslash B$	(<B)	$\mathbf{X}$	$A\backslash C$	$\rightarrow$	$A\backslash B$	$\Rightarrow$	$\mathbf{X}=C\backslash B$	
					$C\backslash B$	$\mathbf{X}$	$\rightarrow$	$A\backslash B$	$\Rightarrow$	$\mathbf{X}=A\backslash C$	

Figure 2: Derivation of inverse combinators

$$P(target=C|R,S)=\max\left\{\begin{array}{r}P(HeadRight|R)\\P(HeadLeft|R)\end{array}\right\}$$

$$P(HeadRight|R)=\left\{\begin{array}{r}P[outside](R)*\\P[inside](S)*\\P(exp=left|R)*\\P(C|R,exp=left)*\\P(S|R,exp=left,C)\end{array}\right\}$$

$$P(HeadLeft|R)=\left\{\begin{array}{r}P[outside](R)*\\P[inside](S)*\\P(exp=right|R)*\\P(S|R,exp=right)*\\P(C|R,exp=right,S)\end{array}\right\}$$

Figure 3: CI probability that the target is category C, given possible categories for result (R) and sister (S).

to attachment errors and ambiguity from adverbials.

The learning step consists in presenting the parser with sentences all of whose words but one are in-lexicon. The parser must have a statistical parsing model, which contains a seed lexicon, a set of CCG combinators, and an optional set of unary and binary rules learned from the training corpus.

First the baseline bottom-up parser is called upon to produce a partial parse chart. The learner takes this partial chart and fills the top right cell with a distribution for the result category based on the end punctuation.[2]

Using this partial chart that contains at least one entry for every leaf cell (except the one OOL target cell) and at least one entry for the result, the learner steps through the chart in a top-down version of CYK (Younger, 1967). For the top-down process, the standard combinators have to be reformulated to take an argument and a result as inputs, rather than two arguments as in the standard bottom-up case. In addition, the learner has access to the non-combinator rules from the parse model, which have been similarly inverted for top-down use. This process continues until the target cell has been filled, and the ranked set of categories is returned.

The probability that the target has a given category is calculated as the greater of the right- or left-headed derivations, according to Figure 3. At training time, the StatOpenCCG parser creates a head-dependency model from the training corpus, in which we can look up the values for the expansion probabilities. Where a value is unavailable, it backs off to a pre-specified value (default 0.0001).[3] The system requires a pruning parameter that limits each cell to the top N most probable categories. Here, we set N=10, to limit the search space and complexity. [4]

Figure 2 sets out the inventory of inverse combinators used in the top-down learning step. Each standard binary CCG combinator motivates two inverse combinators: one for each possible missing item. In the two permissive instances where the sister category's form is the unrestricted B, we limit the sister's valency to 1, in order to keep the learner from generating spurious categories that could result from these two rules being overapplied.

Figure 4 illustrates the workings of the learning

[2]For simple corpora, only S is required, but realistic corpora necessitate a distribution over all result types, including noun phrases and fragments.

[3]This backoff parameter allows adjustment of the expectation of new category types and could be replaced with another smoothing method in subsequent implementations.

[4]Further testing on the McGuffey corpus has shown the average rank of correct tags in the category set to be 1.4.

algorithm for the sentence *The cat* ***X*** *her*. The grey cells are filled as a partial chart by the parser, and the white cells are filled by the top-down learner. Note that taking rule probabilities into account makes the algorithm robust to ambiguity. The highest-ranking lexical category for *her* is $NP[nb]/N$, but the next highest (NP) is preferred in the derivation of the highest-ranking category for the unknown word **X**.

3 Experiment I: Convergence

In the following experiments, we compare Chart Inference to the two baseline methods: Brute Force (BF), derived from Watkinson and Manandhar, and Rule-Based (RB), derived from Yao et al. This section investigates how robust the three systems are to changes in theoriginal seed lexicon.

3.1 Corpus

For this experiment we test the three systems on a reconstructed version of Corpus 1 from Watkinson and Manandhar's experiments.[5] The lexicon contains 40 word-category pairs, including the full stop ($S\backslash S$), which was not in Watkinson's experiment, and one example of noun-verb ambiguity (*saw*). The test sentences are randomly generated from a simple PCFG over the lexicon, and are always presented to the learners in the same order.

3.2 Methods

In order to directly compare the three learning methods, we use the evaluation setting from Watkinson and Manandhar (1999), which consists of a 40-entry target lexicon and a PCFG language model used to randomly generate 1000 sentences. We then specify a seed lexicon and run the learner incrementally, so that it deals with one sentence at a time, then feeds the learned material back into the lexicon. Watkinson's system was shown to fully convergent (they defined convergence as cosine similarity between the seed lexicon ($\vec{S}$) and the target lexicon ($\vec{T}$) exceeding 0.99), whenever the seed lexicon contained at least one instance of each of the category types in the target lexicon (Watkinson and Manandhar, 1999)

[5]The full corpus was not included in any of Watkinson's papers, but its properties were outlined to such an extent that it was straightforward to recreate, though the reconstruction may differ from the original in the distribution of category types. The reconstructed corpus will be released shortly.

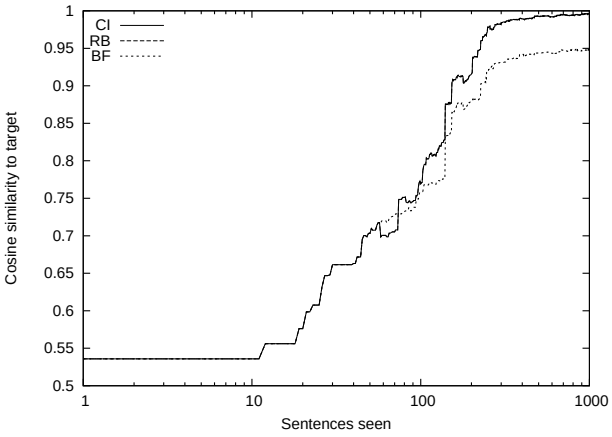


Figure 5: Learning curve for all three methods when the seed contains no ditransitives. CI and RB are identical.

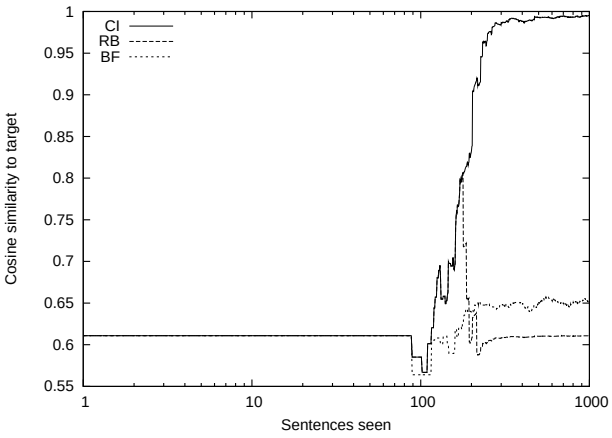


Figure 6: Learning curve for all three methods when seed contains only three determiners and one noun.

3.3 Results

When run incrementally over this toy corpus, both the RB and CI algorithms converge to the target lexicon in an identical sigmoid learning curve (not shown). However, when we start with an impoverished seed, the algorithms' behaviours start to diverge. Figure 5 shows the learning curve for the three methods when the seed lexicon omits all instances of the ditransitive category type $((S\backslash NP)/NP)/NP$. Both RB and CI converge identically as expected, but BF, the lower curve, cannot learn any category types that are not attested in the seed, so it plateaus at 95% similarity.

When the seed is reduced to only three determiners and a noun, CI can still learn the complete

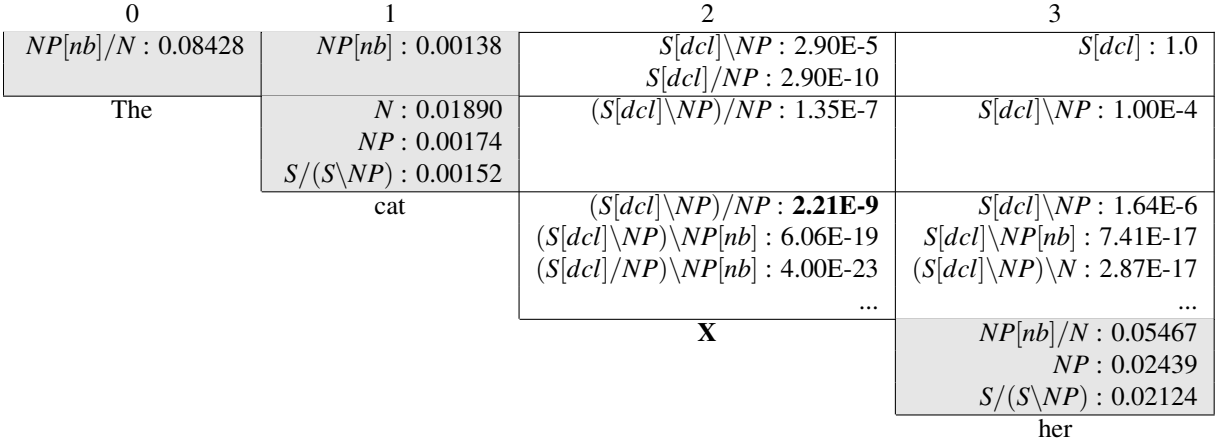


Figure 4: Example of a two-stage derivation using Chart Inference: Grey boxes are filled bottom-up by the partial parser; white boxes top-down by the learner. The target cell (2,2) shows the correct category type as the highest probability solution.

lexicon, despite some initial missteps and a steeper curve. However, the other two methods fail catastrophically (Figure 6). BF never gets going, since it can only correctly learn the remaining nouns. RB is partially successful, but is thwarted by a bad decision at 80% that quickly compounds to diverge from the target lexicon, ending up with higher coverage in the form of more lexical entries, but lower precision, as the final similarity plateaus at the same level as the original seed.

4 Experiments II and III: Coverage

Next, we compare the three learning methods on a larger corpus of natural language, to investigate how well they perform at recovering a wide range of category types in complex settings.

4.1 Corpus

We have constructed a small natural language lexicon based on the first volume of a 6-volume 1836 children's primer, McGuffey's Eclectic Reader.[6] Volume 1 of the McGuffey corpus (MG1) consists of 546 sentences that have been manually annotated with CCG categories, automatically parsed, and then corrected. Volume 2 (MG2) comprises 801 sentences, annotated in the same manner as Volume 1, though not as reliably. The McGuffey corpus makes an ideal seed for development purposes, as it contains a high proportion of simple declarative sentences, but also touches on questions, quotations, passives, and other complex constructions.

[6]The raw text of William Holmes McGuffey's Eclectic Reader is available as an e-book from Project Gutenberg at http://www.gutenberg.org/ebooks/14640. The annotated corpus will be released shortly.

4.2 Methods

In the first of these two experiments we train and test on the same corpus in one pass, attempting to learn each word token in turn and comparing the learned category set to the gold standard annotation. Because we know that the lexicon contains all the necessary entries to correctly parse all the sentences, this addresses the lexical coverage problem discussed in Section 1 of this paper.

The second of these two experiments looks at a more realistic environment for word learning: the parser is initially trained on MG1, then tested on MG2. We evaluate on the gold standard categories in MG2. Since we are not guaranteed to have access to all the necessary word/category pairs in the seed lexicon, the precision and recall values for this second experiment will inevitably be lower than the first.

Figure 7 outlines the process of producing new parsed sentences out of raw text. The process begins like the previous experiment, but then the category set generated by the learner is passed back to the parser, so it can incorporate this new information into its lexicon and produce a full parse. The Hypothesis lexicon is cleared after every sentence.

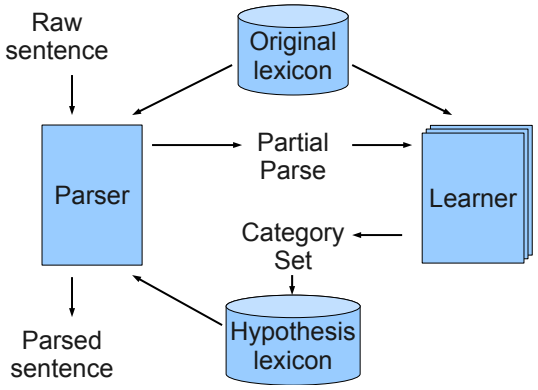


Figure 7: Learning framework for Experiments II-IV.

4.3 Results

Table 1 compares the category match accuracy across the three systems in experiment II, as well as the baseline that chose the the most probable category for the target word's POS. Two tasks are scored: *Top One*, where we evaluate the single highest-scoring category against the gold-standard tag, and *Top Ten*, where we check to see if the gold tag is in the set of the ten highest-probability categories returned by the learner.

CI achieves the best F-scores in both tasks, reaching 76% for *Top One* and 94% for *Top Ten*. POS backoff has an advantage in the *Top Ten* task, especially in recall, since it returns an answer in every case, but CI still outperforms it on F-score. BF achieves the highest precision in the *Top One* task, but takes 30 hours to do so, since it is searching over all possible categories. RB is markedly worse in both precision and recall, but also remarkably fast. CI combines the merits of both BF and RB, yielding a higher F-score than BF and a processing time similar to RB.

In Experiment III, to test the limits of the learners on truly OOL words, we again train on MG1, but test instead on MG2. We can then perform a meaningful error analysis on the results, showing how the three word-learning methods compare in actual practice, in a realistic setting.

Out of its 801 sentences in MG2, only 32 present learning opportunities for the learners, being between 2 and 10 tokens long, containing no internal punctuation or coordination, and containing only one OOL word.

Table 2 shows the category match results of the three systems on MG2. Recall is calculated over the set of learning opportunities, of which there are only 32. BF performs best in all metrics, but the CI results are reasonable. The underlying reason for this behaviour is that the 32 learning targets are all of common categories: over half of them are N or NP. Since the Brute Force learner seeks simply to maximise the tree probability, N and NP are its most common guesses in general.

5 Experiment IV: Domain Adaptation

Clark et al. (2004) identified the problem with using news data to train a parser for a question answering task as the lack of lexical support for question words. Some lexical types were missing entirely. The lexicon for CCGBank §02-21 contains 12 WH-question types, notably lacking some important ones. Clark et al. note the absence of one category in particular: $(S[wq]/(S[dcl]\backslash NP))/N$, the category needed for *What President became Chief Justice after his presidency?*

They attempt to adapt the discriminative C&C parser (Clark and Curran, 2007) to the QA domain by retraining on 500 hand-labelled question sentences, then automatically parsing and hand-correcting an additional 671. The entire set was then used in conjunction with CCGBank §02-21 to train a final parsing model. Their per-word accuracy rose from a 68.5% baseline to 94.6% for the newly trained model.

In this experiment, we examine how close we can get to those results by using Chart Inference to learn WH-question words from the *unlabelled* question corpus. If successful, this would eliminate the human-annotation step for domain adaptation of the kind investigated by (Clark et al., 2004).

5.1 Corpora

We trained the initial parser on the CCGBank (Hockenmaier and Steedman, 2007; Hockenmaier, 2003) training set (§02-21), consisting of 39603 sentences of Wall Street Journal text (Marcus et al., 1993). It is important to note that this training corpus contains only 93 questions in total, so it is not surprising that several category types for question words are entirely unrepresented. It also rein-

	Top One			Top Ten			
	P	R	F	P	R	F	Time (m)
POS	64.91	64.91	64.91	92.55	**92.55**	92.55	1
BF	**80.53**	65.84	72.45	95.97	78.32	86.25	1740
RB	39.77	37.92	38.82	68.46	65.28	66.83	12
CI	78.63	**74.16**	**76.33**	**97.03**	91.52	**94.20**	22

Table 1: Exp. II: Category match results for the three systems on the McGuffey corpus, training and testing on MG1.

	Top One			Top Ten		
	P	R	F	P	R	F
BF	70.83	53.13	60.72	83.33	62.50	71.43
RB	16.13	15.63	15.87	29.03	28.13	28.57
CI	61.90	40.63	49.06	76.19	50.00	60.38

Table 2: Exp. III: Category match results for the three systems on the McGuffey corpus, training on MG1 and testing on MG2. Common categories (N, NP, N/N) are overrepresented in the test data, leading to higher BF scores. POS tags not available for MG2, so no POS baseline is reported.

forces the fact that this is a domain-adaptation task.

We use the same 500-sentence test set as Rimell and Clark (2008b). The test corpus consists of 488 questions, each starting with *What*, *When*, *How*, *Who* or *Where*. The learning corpus contains 1328 questions in a similar distribution.

Only three out of the five categories needed to parse What-questions are present in the CCGBank seed lexicon: $S[wq]/(S[q]/NP)$,[7], $S[wq]/(S[dcl]\backslash NP)$,[8] and $S[wq]/(S[q]/NP)/N$.[9] For this experiment we focus on the subject WH-element extraction category $(S[wq]/(S[dcl]\backslash NP))/N$, as in *Which cat is the grandmother?*. This particular category was chosen as a point of investigation because it is OOL in CCGBank and is common enough to meaningfully evaluate.

5.2 Methods

The baseline is the original StatCCG parser and lexicon. We also employ self-training (Charniak, 1997), in which a parser is used to parse a set of sentences, and then retrained using those output trees. Self-training has had very little success in CCG applications hitherto. McClosky et al (2006) attribute success in self-training to a confluence of circumstances particular to their learning setting, which has the benefit of a discriminative re-ranker, both in the parsing case and in the learning case (McClosky et al., 2008). We follow their recommendations that the best performance is achieved when all the training sentences are parsed at once, rather than incrementally.

We evaluate the success of CI in bootstrapping Wh-question categories from the out-of-domain corpus in two ways. First, we compare the CI output to the gold standard categories labelled in Rimmell and Clark (2008a). Second, we add the parsed questions into the training set, then retrain and finally retest the parser.

The parser was initially trained on CCGBank §02-21 with a word frequency threshold of 5.[10] It produces partial parse charts in the cases where all words in the sentence are in-lexicon, except for the WH-word target, for which the learner attempts to return a category motivated by that context.

We run the learner on the set of 149 sentences from the TREC Question-Answering corpus (Rimell and Clark, 2008b) that contain the word/category pair *What*:$(S[wq]/(S[dcl]\backslash NP))/N$. For this experiment the end-punctuation distribution derived from the training corpus is replaced with a single value: $P(S[wq]|?) = 1$.

[7] Object question category as in *What is the Keystone State?*

[8] Subject question category as in *What lays blue eggs?*

[9] Object WH-element extraction category as in *What continent is Scotland in?*

[10] StatCCG requires a parameter to trade off between training the lexicon and the POS-backoff.

	BL	CI	CI+ST
All Words	84.31	**86.59**	**87.03**
POS=WHQ	53.40	**56.19**	**59.54**
Word=What	55.87	**60.83**	**65.42**
Cat=SubjExt	7.84	**52.94**	**58.82**

Table 3: F-score over individual category matches. Bold means significantly different from the Baseline.

5.3 Results

Table 3 shows the change in F-score throughout this experiment. BL is the baseline condition, where the accuracy is predictably high over all the words in the sentence, but lower when we examine the question words only. It is most telling that the baseline F-score over words that should be tagged with the subject WH-element-extraction category $((S[wq]/(S[dcl]\backslash NP))/N)$ is extremely low. In fact, that seven percent represents only a handful of instances of *Which*, and none of *What*. Applying Chart Inference to the problem results in statistically significant increases in all metrics, but the biggest gain is in the last. When we first apply CI, then self-train over the full training corpus, we further increase all metrics, and again the largest gain is over the target category type specifically. [11] The reason for this can be clearly seen when we evaluate the lexicons created by each method.

Table 4 shows the differences in the impact on the lexicon between baseline (BL), Chart Induction (CI), and the combined method of CI and self-training (CI+ST).[12] CI leaves the initial distribution unchanged while adding seven more category types. One of these is the category we are interested in: $(S[wq]/(S[dcl]\backslash NP))/N$, which is previously associated with *Which* in the baseline lexicon. The other six are spurious categories, and have low counts. Combining the learning mechanisms by running first CI, and then ST, has the effect of introducing the category we need, and then elevating the counts. The probability for $S[wq]$ is elevated as well, as a result of misparses, but the whole process results in better category matches over the test set, as we saw in Table 3.

5.4 Error Analysis

Of the previously known categories, the ST step overwhelmingly prefers three categories: one subject extraction category $S[wq]/(S[dcl]\backslash NP)$ and two object extraction $S[wq]/(S[q]/NP)$ and $(S[wq]/(S[q]/NP))/N$. The remaining categories are classified in Table 4 as either rare (R), spurious (*), or duplicate (D). Rare categories, like $S[wq]$ are used for specialised cases (the sentence *What?*) which occur in PTB, but not in the QA corpus. Spurious categories, like $(S[wq]/PP)/N$ exist in the baseline parser, arising from errors in either the original PTB, or the translation to CCGBank. $S[wq]/S[q]$ is only used where $S[wq]/(S[q]/NP)$ is meant, but fails to capture the extraction. $S[wq]/(S[dcl]/NP)$ is a misinterpretation of sentences requiring $(S[wq]/(S[dcl]\backslash NP))/N$, but without capturing the extracted N.

Five spurious categories are also introduced by the CI learning step. $(S[wq]/S[dcl])/N$ and $(S[wq]/((S[dcl]\backslash NP[expl])/NP))/N$ are spurious forms of $(S[wq]/(S[dcl]\backslash NP))/N$ that arise when the constituent directly right of the target is misparsed; the former misses the extraction and the latter adds an extra dummy subject. $S[wq]/N$ occurs when the main verb of the sentence is treated as a participle, forming a complex nominal argument. $(S[wq]/N)/N$ and $(S[wq]/(S[dcl]/(S[pt]\backslash NP)))/N$ are caused by similar verbal ambiguity.

The classification of $(S[wq]/S[inv])/N$ as a duplicate category is linguistically motivated. Rather than interpret the embedded sentence as declarative, the parser uses *has*:$S[inv]/NP$ to interpret it instead as an inverted sentence. In essence, it cannot see the difference between *What companies have them?* and *What choice have they?* when the NPs lack a case distinction. As such, it duplicates the work of the target $(S[wq]/(S[dcl]\backslash NP))/N$, because the constituents $S[dcl]\backslash NP$ and $S[inv]$ are often synonymous in practice.

As seen in Table 4, the distinction between rare and spurious categories cannot be made on frequency alone, but the best categories are the ones with the highest frequency. Duplicate categories can be considered spurious for the sake of parsing, but

[11] We also ran the experiment using ST only, which performed better than CI alone, but only over a different set consisting entirely of seen categories. We do not report those figures here because they are not commensurable with the CI results.

[12] *What* has 31 categories in total in the baseline lexicon; here we show only the [wq] types.

		$P(W\|C)$			F			$P(C\|W)$		
?	C	BL	CI	CI+ST	BL	CI	CI+ST	BL	CI	CI+ST
R	$S[wq]$	0.09	0.09	0.17	1	1	2	0.006	0.005	0.002
R	$S[wq]/PP$	0.6	0.6	0.6	3	3	3	0.019	0.016	0.003
*	$(S[wq]/PP)/N$	1	1	1	1	1	4	0.006	0.005	0.004
*	$S[wq]/(S[adj]\backslash NP)$	0.5	0.5	0.5	1	1	1	0.006	0.005	0.001
	$S[wq]/(S[dcl]\backslash NP)$	0.37	0.37	0.86	22	22	239	0.137	0.118	0.225
	$S[wq]/(S[dcl]/NP)$	1	1	1	1	1	8	0.006	0.005	0.008
	$(S[wq]/(S[dcl]/NP))/N$	0.5	0.5	0.5	1	1	1	0.006	0.005	0.001
*	$S[wq]/(S[ng]\backslash NP)$	1	1	1	1	1	2	0.006	0.005	0.002
R	$S[wq]/S[poss]$	0.83	0.83	0.83	5	5	5	0.031	0.027	0.005
*	$S[wq]/S[q]$	0.03	0.03	0.12	2	2	9	0.012	0.011	0.008
	$S[wq]/(S[q]/NP)$	0.64	0.64	0.97	16	16	331	0.099	0.086	0.312
	$(S[wq]/(S[q]/NP))/N$	0.36	0.36	0.95	4	4	136	0.025	0.021	0.128
	$(S[wq]/(S[dcl]\backslash NP))/N$	-	**0.5**	**0.96**	-	**4**	**75**	-	**0.021**	**0.071**
*	$S[wq]/N$	-	1	1	-	8	12	-	0.043	0.011
*	$(S[wq]/S[dcl])/N$	-	1	1	-	8	28	-	0.043	0.026
*	$(S[wq]/N)/N$	-	1	1	-	4	7	-	0.021	0.007
D	$(S[wq]/S[inv])/N$	-	1	1	-	3	78	-	0.016	0.074
*	$(S[wq]/(S[dcl]/(S[pt]\backslash NP)))/N$	-	1	1	-	1	2	-	0.005	0.002
*	$(S[wq]/((S[dcl]\backslash NP[expl])/NP))/N$	-	1	1	-	1	12	-	0.005	0.011

Table 4: Exp. IV: Lexical category distribution for the word *What* in the baseline §02-21 of CCGBank (BL), after Chart Inference (CI), and after first applying Chart Inference, then self-training (CI+ST). Column 1 classifies low-frequency categories as rare (R), spurious (*) or duplicate (D). Cateogories above the middle line are present in the Baseline lexicon; below are induced.

are linguistically interesting, and if they are frequent enough, that is possibly an indication that the structure of the lexicon or the grammar is non-optimal.

6 Conclusion and Future Work

Chart Inference is a useful tool for finding OOL categories. It has been shown to outperform both the brute-force and rule-based systems. When used in conjunction with self-training, CI presents a valuable framework for domain adaptation in the case where whole category types are missing from the lexicon.

It remains to put Chart Inference into an appropriate framework for improving coverage over the baseline WSJ-trained StatCCG parser. We estimate an upper bound of 20% error reduction possible over CCGBank §00, if the lexicon is expanded to cover all the necessary word/category pairs. Improving global F-score for §23 is of course very difficult. The lexical entries CI finds are by definition rare and at the scale we are running, they are unlikely to occur in those 2000 sentences. We believe our analysis of the lexical items themselves shows that we are learning a high proportion of good lexical entries.

The problem of discovering missing categories for known words remains. We have shown through adapting to the question domain that it is possible to make focused improvements when we can identify the gaps in coverage (as in wh-question words), but in order to address the challenge of automatic lexicon extension fully, quite different techniques for generalising lexical entries for seen words will be require.

Acknowledgements

The authors would like to thank Steve Clark and Laura Rimell for provision of the annotated QA corpus, Christos Christodoulopoulos for the StatOpenCCG parser, Tejaswini Deoskar for editorial advice, and Luke Zettlemoyer for considerable mathematical assistance. This work was partially funded by EU ERC Advanced Fellowship 249520 GRAMPLUS, IST Cognitive Systems IP EC-FP7-270273 "XPERIENCE" and a grant from Wolfson Microelectronics.

References

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI '97*, pages 598–603.

Christos Christodoulopoulos. 2008. Creating a natural logic inference system with combinatory categorial grammar. Master's thesis, School of Informatics, University of Edinburgh.

Stephen Clark and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of EMNLP '03*, pages 97–104, Morristown, NJ, USA.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark, Mark Steedman, and James Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of EMNLP '04*, pages 111–118.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Julia Hockenmaier. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Tsuneaki Kato. 1994. Yet another chart-based technique for parsing ill-formed input. In *Proceedings of ANLC '94*, pages 107–112, Stroudsburg, PA, USA. Association for Computational Linguistics.

D. Kazakov, S. Pulman, and S. Muggleton. 1998. The FraCas dataset and the LLL challenge. Technical report, SRI International.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330, June.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of NAACL-HLT '06*, pages 152–159.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *Proceedings of COLING '08*, pages 561–568, Morristown, NJ, USA.

Chris S. Mellish. 1989. Some chart based techniques for parsing ill-formed input. In *Proceedings of the ACL '89*, pages 102–109, Morristown, NJ, USA. Association for Computational Linguistics.

Laura Rimell and Stephen Clark. 2008a. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of EMNLP '08*, pages 475–484, Stroudsburg, PA, USA.

Laura Rimell and Stephen Clark. 2008b. Constructing a parser evaluation scheme. In *COLING '08: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 44–50, Stroudsburg, PA, USA.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Stephen Watkinson and Suresh Manandhar. 1999. Unsupervised lexical learning of categorial grammars. In *ACL'99: Workshop in Unsupervised Learning in Natural Language Processing*.

Stephen Watkinson and Suresh Manandhar. 2000. Unsupervised lexical learning with categorial grammars using the LLL corpus. In James Cussens and Savso Dvzeroski, editors, *Learning Language in Logic*, volume 1925 of *Lecture Notes in Artificial Intelligence*. Springer.

Stephen Watkinson and Suresh Manandhar. 2001a. Acquisition of large scale categorial grammar lexicons. In *Proceedings of PACLING '01*.

Stephen Watkinson and Suresh Manandhar. 2001b. A psychologically plausible and computationally effective approach to learning syntax. In Walter Daelemans and R'emi Zajac, editors, *Proceedings of CoNLL '01)*, pages 160 – 167.

Michael White and Jason Baldridge. 2003. Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation*, pages 119–126.

Xuchen Yao, Jianqiang Ma, Sergio Duarte, and Cagri Coltekin. 2009a. An inference-rules based categorial grammar learner for simulating language acquisition. In *Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning*, Tillburg.

Xuchen Yao, Jianqiang Ma, Sergio Duarte, and Cagri Coltekin. 2009b. Unsupervised syntax learning with categorial grammars using inference rules. In *Proceedings of The 14th Student Session of the European Summer School for Logic, Language, and Information*, Bordeaux.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

A Fast, Accurate, Non-Projective, Semantically-Enriched Parser

Stephen Tratz and **Eduard Hovy**
Information Sciences Institute
University of Southern California
Marina del Rey, California 90292
{stratz,hovy}@isi.edu

Abstract

Dependency parsers are critical components within many NLP systems. However, currently available dependency parsers each exhibit at least one of several weaknesses, including high running time, limited accuracy, vague dependency labels, and lack of non-projectivity support. Furthermore, no commonly used parser provides additional shallow semantic interpretation, such as preposition sense disambiguation and noun compound interpretation. In this paper, we present a new dependency-tree conversion of the Penn Treebank along with its associated fine-grain dependency labels and a fast, accurate parser trained on it. We explain how a non-projective extension to shift-reduce parsing can be incorporated into non-directional easy-first parsing. The parser performs well when evaluated on the standard test section of the Penn Treebank, outperforming several popular open source dependency parsers; it is, to the best of our knowledge, the first dependency parser capable of parsing more than 75 sentences per second at over 93% accuracy.

1 Introduction

Parsers are critical components within many natural language processing (NLP) systems, including systems for information extraction, question answering, machine translation, recognition of textual entailment, summarization, and many others. Unfortunately, currently available dependency parsers suffer from at least one of several weaknesses including high running time, limited accuracy, vague dependency labels, and lack of non-projectivity support. Furthermore, few parsers include any sort of additional semantic interpretation, such as interpretations for prepositions, possessives, or noun compounds.

In this paper, we describe 1) a new dependency conversion (Section 3) of the Penn Treebank (Marcus, et al., 1993) along with the associated dependency label scheme, which is based upon the Stanford parser's popular scheme (de Marneffe and Manning, 2008), and a fast, accurate dependency parser with non-projectivity support (Section 4) and additional integrated semantic annotation modules for automatic preposition sense disambiguation and noun compound interpretation (Section 5). We show how Nivre's (2009) swap-based reordering technique for non-projective shift-reduce-style parsing can be integrated into the non-directional easy-first framework of Goldberg and Elhadad (2010) to support non-projectivity, and we report the results of our parsing experiments on the standard test section of the PTB, providing comparisons with several freely available parsers, including Goldberg and Elhadad's (2010) implementation, MALTPARSER (Nivre et al., 2006), MSTPARSER (McDonald et al., 2005; McDonald and Pereira, 2006), the Charniak (2000) parser, and the Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007).

The experimental results show that the parser is substantially more accurate than Goldberg and Elhadad's original implementation, with fairly similar overall speed. Furthermore, the results prove that Stanford-granularity dependency labels can be learned by modern dependency parsing systems when using our Treebank conversion, unlike the Stanford conversion, for which Cer et al. (2010) show that this isn't the case.

The optional semantic annotation modules also

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1257–1268,
Edinburgh, Scotland, UK, July 27–31, 2011.

perform well, with the preposition sense disambiguation module exceeding the accuracy of the previous best reported result for fine-grained preposition sense disambiguation (85.7% vs Hovy et al.'s (2010) 84.8%), the possessives interpretation system achieving over 85% accuracy, and the noun compound interpretation system performing similarly to an earlier version described by Tratz and Hovy (2010) at just over 79% accuracy.

2 Background

The NLP community has recently seen a surge of interest in dependency parsing, with several CoNLL shared tasks focusing on it (Buchholz and Marsi, 2006; Nivre et al., 2007). One of the main advantages of dependency parsing is the relative ease with which it can handle non-projectivity[1]. Additionally, since each word is linked directly to its head via a link that, ideally, indicates the syntactic dependency type, there is no difficulty in determining either the syntactic head of a particular word or the syntactic relation type, whereas these issues often arise when dealing with constituent parses[2].

Unfortunately, most currently available dependency parsers produce relatively vague labels or, in many cases, produce no labels at all. While the Stanford fine-grain dependency scheme (de Marneffe and Manning, 2008) has proven to be popular, recent experiments by Cer et al. (2010) using the Stanford conversion of the Penn Treebank indicate that it is difficult for current dependency parsers to learn. Indeed, the highest scoring parsers trained using the MSTPARSER (McDonald and Pereira, 2006) and MALTPARSER (Nivre et al., 2006) parsing suites achieved only 78.8 and 81.1 labeled attachment F_1, respectively. This contrasted with the much higher performance obtained using a constituent-to-dependency conversion approach with accurate, but much slower, constituency parsers such as the Charniak and Johnson (2005) and Berkeley (Petrov et al., 2006; Petrov and Klein, 2007) parsers, which achieved 89.1 and 87.9 labeled F_1 scores, respectively.

[1] A tree is non-projective if the sequence of words visited in a left-to-right, depth-first traversal of the sentence's parse tree is different than the actual word order of the sentence.

[2] These latter two issues are not problems for constituent parses with binarized output and functional tags.

Though there are many syntactic parsers than can reconstruct the grammatical structure of a text, there are few, if any, accurate and widely accepted systems that also produce shallow semantic analysis of the text. For example, a parser may indicate that, in the case of 'ice statue', 'ice' modifies 'statue' but will not indicate that 'ice' is the *substance* of the statue. Similarly, a parser will indicate which words a preposition connects but will not give any semantic interpretation (e.g., 'the boy *with* the pirate hat' $\rightarrow$ *wearing or carrying*, 'wash *with* cold water' $\rightarrow$ *means*, 'shave *with* the grain' $\rightarrow$ *in the same direction as*). While, in some cases, it may be possible to use the output from a separate system for this purpose, doing so is often difficult in practice due to a wide variety of complications, including programming language differences, alternative data formats, and, sometimes, other parsers.

3 Dependency Conversion

3.1 Relations and Structure

Most recent English dependency parsers produce one of three sets of dependency types: unlabeled, some variant of the coarse labels used by the CoNLL dependency parsing shared-tasks (Buchholz and Marsi, 2006; Nivre et al., 2007) (e.g., ADV, NMOD, PMOD), or Stanford's dependency labels (de Marneffe and Manning, 2008). Unlabeled dependencies are clearly too impoverished for many tasks. Similarly, the coarse labels of the CoNLL tasks are not very specific; for example, the same relation, NMOD, is used for determiners, adjectives, nouns, participle modifiers, relative clauses, etc. that modify nouns. In contrast, the Stanford relations provide a more reasonable level of granularity.

Our dependency relation scheme is similar to Stanford's basic scheme but has several differences. It introduces several new relations including *ccinit* "initial coordinating conjunction", *cleft* "cleft clause", *combo* "combined term", *extr* "extraposed element", *infmark* "infinitive marker 'to' ", *objcomp* "object complement", *postloc* "post-modifying location", *sccomp* "clausal complement of 'so' ", *vch* "verbal chain" and *whadvmod* "wh- adverbial modifier". The *nsubjpass*, *csubjpass*, and *auxpass* relations of Stanford's are left out because adding them up front makes learning more difficult and the fact

abbrev	abbreviation	*csubjpass*	clausal subject (passive)	pobj	prepositional object
acomp	adjectival complement	det	determiner	poss	possessive
advcl	adverbial clause	dobj	direct object	possessive	possessive marker
advmod	adverbial modifier	**extr**	extraposed element	**postloc**	post-modifying location
agent	'by' agent	expl	'there' expletive	preconj	pre conjunct
amod	adjectival modifier	**infmark**	infinitive marker ('to')	predet	predeterminer
appos	appositive	infmod	infinite modifier	prep	preposition
attr	attributive	iobj	indirect object	prt	particle
aux	auxillary	mark	subordinate clause marker	punct	punctuation
auxpass	auxillary (passive)	measure	measure modifier	purpcl	purpose clause
cleft	cleft clause	neg	negative	quantmod	quantifier modifier
cc	coordination	nn	noun compound	rcmod	relative clause
ccinit	initial CC	nsubj	nominal subject	*rel*	relative
ccomp	clausal complement	*nsubjpass*	nominal subject (passive)	**sccomp**	clausal complement of 'so'
combo	combination term	num	numeric modifier	tmod	temporal modifier
compl	complementizer	number	compound number	**vch**	verbal chain
conj	conjunction	**objcomp**	object complement	**whadvmod**	wh- adverbial
cop	copula complement	parataxis	parataxis	xcomp	clausal complement w/o subj
csubj	clausal subject	partmod	participle modifier		

Table 1: Dependency scheme with differences versus basic Stanford dependencies highlighted. Bold indicates the relation does not exist in the Stanford scheme. Italics indicate the relation appears in Stanford's scheme but not ours.

that a *nsubj*, *csubj*, or *aux* is passive can easily be determined from the final tree. Stanford's *aux* dependencies are replaced using verbal chain (*vch*) links; conversion of these to Stanford-style *aux* dependencies is also trivial as a post-processing step.[3] The *attr* dependency is excluded because it is redundant with the *cop* relation due to different handling of copula, and the dependency scheme does not have an *abbrev* label because this information is not provided by the Penn Treebank. The dependency scheme with differences with Stanford highlighted is presented in Table 1.

In addition to using a slightly different set of dependency names, a handful of relations, notably *cop*, *conj*, and *cc*, are treated in a different manner. These differences are illustrated by Figure 1. The Stanford scheme's treatment of copula may be one reason why dependency parsers have trouble learning and applying it. Normally, the head of the clause is a verb, but, under Stanford's scheme, if the verb happens to be a copula, the complement of the copula (*cop*) is treated as the head of the clause instead.

[3]The parsing system includes an optional script that can convert *vch* arcs into *aux* and *auxpass* and the subject relations into *csubjpass* and *nsubjpass*.

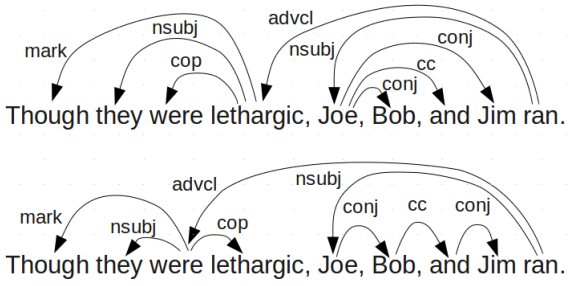


Figure 1: Example comparing Stanford's (top) handling of copula and coordinating conjunctions with ours (bottom).

3.2 Conversion Process

A three-step process is used to convert the Penn Treebank (Marcus, et al., 1993) from constituent parses into dependency trees labeled according to the dependency scheme presented in the prior section. The first step is to apply the noun phrase structure patch created by Vadas and Curran (2007), which adds structure to the otherwise flat noun phrases (NPs) of the Penn Treebank (e.g., '(metal soup pot cover)' would become '(metal (soup pot) cover)'). The second step is to apply a version of Johansson and Nugues' (2007) constituent-to-dependency converter with some head-finding rule modifications; these rules, with changes highlighted

(WH)?NP|NX|NML|NAC	**FW|NML**|NN* JJR $|# CD|**FW QP** JJ|**NAC** JJS **PRP ADJP RB[SR] VBG|DT|WP**
	RB NP-ϵ **S|SBAR|UCP|PP SINV|SBARQ|SQ UH VP|NP VB|VBP**
ADJP|JJP	NNS QP NN $|# JJ VBN VBG (AD|J)JP **ADVP** JJR NP|NML JJS DT FW RBR RBS SBAR RB
ADVP	**RB|RBR|JJ|JJR** RBS FW ADVP TO CD IN NP|NML JJS NN
PRN	S* **VP NN*|NX|NML NP** W* PP|IN ADJP|JJ ADVP RB **NAC VP INTJ**
QP	$|# NNS NN **CD** JJ RB DT NCD QP IN **CC** JJR JJS
SBARQ	SQ S SBARQ **SINV** FRAG
SQ	VBZ VBD VBP VB MD *-PRD SQ VP **FRAG X**
UCP	**[QNVP]P|S*|UCP|NML|PR[NT]|RRC|NX|NAC|FRAG|INTJ|AD[JV]P|LST|WH*|X**
VP	VBD|AUX VBN MD VBZ VB VBG VBP VP **POS** *-PRD ADJP JJ NN NNS NP|NML
WHADJP	CC **JJ** WRB ADJP
WHADVP	CC WRB|**RB**
X	**[QNVP]P|S*|UCP|NML|PR[NT]|RRC|NX|NAC|FRAG|INTJ|AD[JV]P|LST|WH*|X|CONJP**
LST	LS **: DT|NN|SYM**

Figure 2: Modified head-finding rules. Underline indicates that the search is performed in a left-to-right fashion instead of the default right-to-left order. NML and JJP are both products of Vadas and Curran's (2007) patch. Bold indicates an added or moved element; for the original rules, see the paper by Johansson and Nugues (2007).

in bold, are provided in Figure 2. Finally, an additional script makes additional changes and converts the intermediate output into the dependency scheme.

This dependency conversion has several advantages to it. Using the modified head-finding rules for Johansson and Nugues' (2007) converter results in fewer buggy trees than were present in the CoNLL shared tasks, including fewer trees in which words are headed by punctuation marks. For sections 2–21, there are far fewer generic *dep/DEP* relations (2,765) than with the Stanford conversion (34,134) or the CoNLL 2008 shared task conversion (23,811). Also, the additional conversion script contains various rules for correcting part-of-speech (POS) errors using the syntactic structure as well as additional rules for some specific word forms, mostly common words with inconsistent taggings. Many of these changes cover part-of-speech problems discussed by Manning (2011), including VBD/VBN, VBZ/NNS, NNP/NNPS, and IN/WDT/DT issues. In total, the script changes over 9,500 part-of-speech tags, with the most common change being to change preposition tags (IN) into adverb tags (RB) for cases where there is no prepositional complement/object. The top fifteen of these changes are presented in Table 2. The conversion script contains a variety of additional rules for modifying the parse structure and fixing erroneous trees as well, including cases where one or more POS tags were incorrect and, as such, the initial dependency parse was flawed. Quick manual inspections of the changes suggested that the vast majority are accurate.

In the final output from the conversion, the number of sentences with one or more words dependent on non-projective arcs in sections 2–21 is 3,245—about 8.1% of the dataset. About 1.3% of this, or 556 of sentences, is due to the secondary conversion script, with sentences containing approximate currency amounts (e.g., *about $ 10*) comprising the bulk of difference. For these, the quantifying text (e.g., *about*, *over*, *nearly*), is linked to the number following the currency symbol instead of to the currency symbol as it was in the CoNLL 2008 task.

Original	New	# of changes
IN	RB	1128
JJ	NN	787
VBD	VBN	601
RB	IN	462
VBN	VBD	441
NN	JJ	409
NNPS	NNP	405
IN	WDT	388
VBG	NN	223
DT	IN	220
RB	JJ	214
VB	VBP	184
NN	NNS	169
RB	NN	157
NNS	VBZ	148

Table 2: Top 15 part-of-speech tag changes performed by the conversion script.

4 Parser

4.1 Algorithm

The parsing approach is based upon the *non-directional easy-first* algorithm recently presented by Goldberg and Elhadad (2010). Their original algorithm behaves as follows. For a sentence of length n, the algorithm performs a total of n steps. In each step, one of the unattached tokens is added as a child to one of its current neighbors and is then removed from the list of unprocessed tokens. When only one token remains unprocessed, it is designated as the root. Provided that only a constant number of potential attachments need to be re-evaluated after each step, which is the case if one restricts the context for feature generation to a constant number of neighboring tokens, the algorithm can be implemented to run in O($n \log n$). However, since only O(n) dot products must be calculated by the parser and these have a large constant associated with them, the running time will rival O(n) parsers for any reasonable n, and, thus, a naive O(n^2) implementation will be nearly as fast as a priority queue implementation in practice.[4]

The algorithm has a couple potential advantages over standard shift-reduce style parsing algorithms. The first advantage is that performing easy actions first may make the originally difficult decisions easier. The second advantage is that performing parse actions in a more flexible order than left-to-right/right-to-left shift-reduce parsing reduces the chance of error propagation.

Unfortunately, the original algorithm does not support non-projective trees. To extend the algorithm to support non-projective trees, we introduce *move-right* and *move-left* operations similar to the stack-to-buffer *swaps* proposed by Nivre (2009) for shift-reduce style parsing. Thus, instead of attaching a token to one of its neighbors at each step, the algorithm may instead decide to *move* a token past one of its neighbors. Provided that no node is allowed to be moved past a token in such a way that a previous *move* operation is undone, there can be at most O(n^2) *moves* and the overall worst-case complexity becomes O($n^2 \log n$). While theoretically slower, this has a limited impact upon actual parsing times in practice, especially for languages with relatively fixed word order such as English.[5] Though Goldberg and Elhadad's (2010) original implementation only supports unlabeled dependencies, the algorithm itself is in no way limited in this regard, and it is simple enough to add labeled dependency support by treating each dependency label as a specific type of *attach* operation (e.g., *attach_as_nsubj*), which is the method used by this implementation. Pseudocode for the non-directional easy-first algorithm with non-projective support is given in Algorithm 1.

```
input : w_1 ... w_n, #the sentence
        m, #the model
        k, #the context width
        actions, #the list of parse actions
        φ, #the feature generator
output: tree #a collection of dependency arcs
words = copyOf(s);
stale = copyOf(s);
cache; #cache of action scores
while |words| > 1 do
    for w ∈ stale do
        for act ∈ actions do
            cache[w,act] = score(act, φ(w,...),
            m);
        stale.remove(w);
    best = argmax_{a∈actions&valid(a),w∈words} cache[w, a]
    if isMove(best) then
        i =
            words.index(getTokenToMove(best));
        words.move(i, isMoveLeft(best) ? -1
        : 1);
    else
        arc = createArc(best);
        tree.add(arc);
        i = words.index(getChild(arc));
        words.remove(i);
    for x ∈ -k,...,k do
        stale.add(words.get(index+x));
return tree
```

Algorithm 1: Modified version of Goldberg and Elhadad's (2010) Easy-First Algorithm with non-projective support.

[4] See Goldberg and Elhadad (2010) for more explanation.

[5] See Nivre (2009) for more information on the effect of reordering operations on parse time.

4.2 Features

One of the key aspects of the parser is the complex set of features used. The feature set is based off the features used by Goldberg and Elhadad (2010) but has a significant number of extensions. Various feature templates are specifically designed to produce features that help with several syntactic issues including preposition attachment, coordination, adverbial clauses, clausal complements, and relative clauses. Unfortunately, there is insufficient space in this paper to describe them all here. However, a list of feature templates will be provided with the parser download.

Several of the feature templates use unsupervised word clusters created with the Brown et al. (1992) hierarchical clustering algorithm. The use of this algorithm was inspired by Koo et al. (2008), who used the top branches of the cluster hierarchy as features. However, unlike Koo et al.'s (2008) parser, the fine-grained cluster identifiers are used instead of just the top 4-6 branches of the cluster hierarchy. The 175 word clusters utilized by the parser were created from the New York Times corpus (Sandhaus, 2008). Some examples from the clusters are presented in Figure 3. The ideal number of such clusters was not thoroughly investigated.

while where when although despite unless unlike ...
why what whom whatever whoever whomever whence ...
based died involved runs ended lived charged born ...
them him me us himself themselves herself myself ...
really just almost nearly simply quite fully virtually ...
know think thought feel believe knew felt hope mean ...
into through on onto atop astride Saturday/Early thru ...
Ms. Mr. Dr. Mrs. Judge Miss Professor Officer Colonel ...
John President David J. St. Robert Michael James George ...
wife own husband brother sister grandfather beloved ...
often now once recently sometimes clearly apparently ...
everyone it everybody somebody anybody nobody hers ...
around over under among near behind outside across ...
Clinton Bush Johnson Smith Brown Williams King ...
children companies women people men things students ...

Figure 3: High frequency examples from 15 of the Brown clusters.

4.3 Training

The parsing model is trained using a variant of the structured perceptron training algorithm used in the original Goldberg and Elhadad (2010) implementation. The general idea of the algorithm is to iterate over the sentences and, whenever the model predicts an incorrect action, update the model weights. Following Goldberg and Elhadad, parameter averaging is used to reduce overfitting.

Our implementation varies slightly from that of Goldberg and Elhadad (2010). The difference is that, at any particular step for a given sentence, the algorithm continues to update the weight vector as long as *any* invalid action is scored higher than *any* valid action, not just the highest scoring valid action; unfortunately, this change significantly slowed down the training process. In early experiments, this change produced a slight improvement in accuracy though it also slowed training significantly. In later experiments using additional feature templates, this change ceased to have any notable impact on the overall accuracy, but it was kept anyway. [6]

The oracle used to determine whether a *move* operation should be considered legal during the training phase is similar to Nivre et al.'s (2009) improved oracle based upon *maximal projective subcomponents*. As an additional restriction, during training, *move* actions were only considered valid either if no other action was valid or if the token to be moved already had all its children attached and moving it caused it to be adjacent to its parent. This fits with Nivre et al.'s (2009) intuition that it is best to delay word reordering as long as possible.

4.4 Speed Enhancements

To enhance the speed for practical use, the parser uses constraints based upon the part-of-speech tags of the adjacent word pairs to eliminate invalid dependencies from even being evaluated. A relation is only considered between a pair of words if such a relation was observed in the training data between a pair of words with the same parts-of-speech (with the exception of the generic *dep* dependency, which is permitted between any POS tag pair). Early experiments utilizing similar constraints showed an improvement in parsing speed of about 16% with no significant impact on accuracy, regardless of whether the constraints were enforced during training.

[6] See Goldberg and Elhadad (2010) for more description of the general training procedure.

System	Arc Accuracy		Perfect Sentences		Non-Proj Arcs	
	Labeled	Unlabeled	Labeled	Unlabeled	Labeled	Unlabeled
THIS WORK	**92.1 (93.3)**	**93.7 (94.3)**	**38.4 (42.5)**	**46.2 (48.5)**	66.5 (69.7)	**69.3** (71.7)
THIS WORK$_{no\ clusters}$	91.8 (93.1)	93.4 (94.1)	38.2 (42.3)	45.5 (47.3)	**67.3 (70.9)**	**69.3 (72.5)**
THIS WORK$_{moves\ disabled}$	91.7 (92.9)	93.3 (93.9)	37.1 (40.8)	44.2 (46.2)	21.1 (21.1)	22.7 (21.9)
NON-DIR EASY FIRST	*	91.2 (92.0)	*	37.8 (39.4)	*	15.1 (16.3)
EISNER$^{\dagger}{}_{MST}$	90.9 (92.2)	92.8 (93.5)	32.1 (35.6)	40.6 (42.3)	62.5 (65.3)	63.7 (66.9)
CHU-LIU-EDMONDS$_{MST}$	90.0 (91.2)	91.8 (92.5)	28.4 (31.3)	35.0 (36.4)	62.9 (65.3)	64.1 (66.5)
ARC-EAGER$_{Malt}$	89.8 (91.1)	91.3 (92.1)	31.6 (34.2)	37.4 (38.5)	19.5 (19.5)	20.3 (19.9)
ARC-STANDARD$_{Malt}$	88.3 (89.5)	89.7 (90.4)	31.4 (34.1)	36.1 (37.3)	13.1 (12.0)	13.9 (12.7)
STACK-EAGER$_{Malt}$	90.0 (91.2)	91.5 (92.3)	34.5 (37.5)	40.4 (41.9)	51.8 (53.8)	53.8 (55.4)
STACK-LAZY$_{Malt}$	90.4 (91.7)	91.9 (92.8)	34.8 (37.7)	40.6 (42.5)	61.8 (63.3)	63.3 (65.3)
CHARNIAK‡	*	93.2	*	43.5	*	32.3
BERKELEY‡	*	93.3	*	43.6	*	34.3

Table 3: Parsing results for section 23 of the Penn Treebank (punctuation excluded). Results in parentheses were produced using gold POS tags. †Eisner (1996) algorithm with non-projective rewriting and second order features. ‡Results not directly comparable; see text. *Labeled dependencies not available/comparable.

4.5 Evaluation

The following split of the Penn Treebank (Marcus, et al., 1993) was used for the experiments: sections 2–21 for training, 22 for development, and 23 for testing.

For part-of-speech (POS) tagging, we used an in-house SVM-based POS tagger modeled after the work of Giménez and Márquez (2004) [7]. The training data was tagged in a 10-fold fashion; each fold was tagged using a tagger trained from the nine remaining folds. The development and test sections were tagged by an instance of the tagger trained using the entire training set. The full details of the POS tagger are outside the scope of this paper; it is included with the parser download.

The final parser was trained for 31 iterations, which is the point at which its performance on the development set peaked. One test run was performed with non-projectivity support disabled in order to get some idea of the impact of the *move* operations on the parser's overall performance; also, since the parsers used for comparison had no access to the unsupervised word clusters, an additional instance of the parser was trained with every word treated as belonging to the same cluster so as to facilitate a more fair comparison.

Seven different dependency parsing models were trained for comparison using the following open source parsing packages: Goldberg and Elhadad's (2010)'s *non-directional easy-first* parser, MALTPARSER (Nivre et al., 2006), and MSTPARSER (McDonald and Pereira, 2006)[8]. The model trained using Goldberg and Elhadad's (2010) easy-first parser serves as something of a baseline. The four MALTPARSER parsing models used the *arc-eager*, *arc-standard*, *stack-eager*, and *stack-lazy* algorithms. One of the MSTPARSER models used the Chu-Liu-Edmonds maximum spanning tree approach, and the other used the Eisner (1996) algorithm with second order features and a non-projective rewriting post-processing step.

Unfortunately, it is not possible to directly compare the parser's accuracy with most popular constituent parsers such as the Charniak (2000) and Berkeley (Petrov et al., 2006; Petrov and Klein, 2007) parsers[9] both because they do not produce functional tags for subjects, direct objects, etc., which are required for the final script of the constituent-to-dependency conversion routine, and because they determine part-of-speech tags in conjunction with the parsing. However, it is possible to compute approximate unlabeled accuracy scores by training the constituent parsers on the NP-patched (Vadas and Curran, 2007) version of the data and then running the test output through just the first conversion script—that is, the modified version of Johansson and Nugues' (2007) converter.

[7] 97.42% accuracy on traditional POS evaluation (Penn Treebank WSJ sections 22-24).

The results of the experiment are given in Table 3, including accuracy for individual arcs, non-projective arcs only, and full sentence match. Punctuation is excluded in all the result computations. To determine whether an arc is non-projective, the following heuristic was used. Traverse the sentence in a depth-first search, starting from the imaginary root node and pursuing child arcs in order of increasing absolute distance from their parent. Whenever an arc being traversed is found to cross a previously traversed arc, mark it as non-projective and continue. To evaluate the impact of part-of-speech tagging error, results for parsing using the gold standard part-of-speech tags are also included.

We also measured the speed of the parser on the various sentences in the test collection. For reasonable sentence lengths, the parser scales quite well. The scatterplot depicting the relation between sentence length and parsing time is presented in Figure 5.

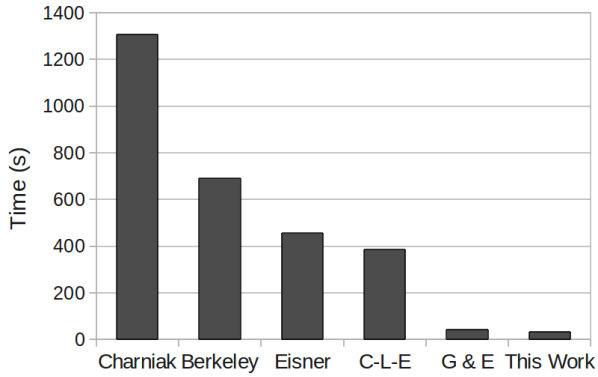


Figure 4: Parse times for Penn Treebank section 23 for the parsers on a PC with a 2.4Ghz Q6600 processor and 8GB RAM. MALTPARSER ran substantially slower than the others, perhaps due to its use of polynomial kernels, and isn't shown. (C-L-E - Chu-Liu-Edmonds, G&E - Goldberg and Elhadad (2010)).

4.5.1 Results Discussion

The parser achieves 92.1% labeled and 93.7% unlabeled accuracy on the evaluation, a solid result and about 2.5% higher than the original easy-first implementation of Goldberg and Elhadad (2010). Furthermore, the parser processed the entire test section in just over 30 seconds—a rate of over 75 sentences per second, substantially faster than most of the other parsers.

Not surprisingly, the results for non-projective arcs are substantially lower than the results for all arcs, and the systems that are designed to handle them outperformed the strictly projective parsers in this regard.

The negative effect of part-of-speech tagging error appears to impact the different parsers about the same amount, with a loss of .6% to .8% in unlabeled accuracy and 1.1% to 1.3% in labeled accuracy.

The 93.2% and 93.3% accuracy scores achieved by the Charniak and Berkeley parsers are not too different from the 93.7% result, but, of course, it is important to remember that these scores are not directly comparable.

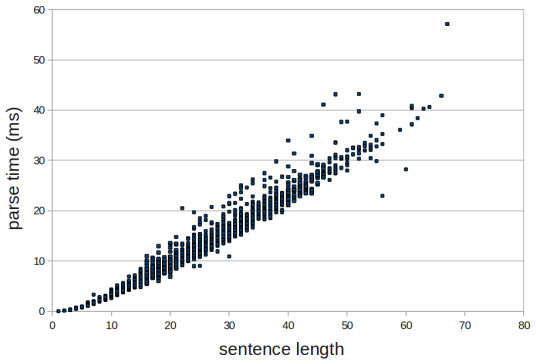


Figure 5: Sentence length versus parse time. Median times for five runs over section 23.

5 Shallow Semantic Annotation

To create a more informative parse, the parser includes four optional modules, a preposition sense disambiguation (PSD) system, a work-in-progress *'s*-possessive interpretation system, a noun compound interpretation system, and a PropBank-based semantic role labeling system[10]. Taken together, these integrated modules enable the parsing system to produce substantially more informative output than a traditional parser.

Preposition Sense Disambiguation The PSD system is a newer version of the system described

[8] Versions 1.4.1, 0.4.3b, and 0.2, respectively

[9] Versions 1.1 and 05Aug16, respectively

[10] Lack of space prohibits a sufficiently thorough discussion of these individual components and their evaluations, but additional information will be available with the system download.

by Tratz and Hovy (2009) and Hovy et al. (2010); it achieves 85.7% accuracy on the SemEval-2007 fine-grain PSD task (Litkowski and Hargraves, 2007), which is a statistically significant ($p<=0.05$; upper-tailed z test) increase over the previous best reported result for this dataset, Hovy et al.'s (2010) 84.8%.

Noun Compound Interpretation The noun compound interpretation system is a newer version of the system described by Tratz and Hovy (2010) with similar accuracy (79.6% vs 79.3% using 10-fold cross-validation[11]).

Possessives Interpretation The possessive interpretation system assigns interpretations to *'s* possessives (e.g., John's arm → PART-OF, Mowgli's capture → PATIENT/THEME). The current system achieves over 85.0% accuracy, but it is important to note that the annotation scheme, automatic classifier, and dataset are all still under active development.

PropBank SRL The PropBank-based semantic role labeling system achieves 86.8 combined F_1 measure for automatically-generated parse trees calculated over both predicate disambiguation and argument/adjunct classification (89.5 F_1 on predicate disambiguation, 85.6 F_1 on argument and adjuncts corresponding to dependency links, and 86.8 F_1); this score is not directly comparable to any previous work due to some differences, including differences in both the parse tree conversion and the PropBank conversion. The most similar work is that of the CoNLL shared task work (Surdeanu et al., 2008; Hajič et al., 2009).

6 Related Work

Non-projectivity. There are two main approaches used in recent NLP literature for handling non-projectivity in parse trees. The first is to use an algorithm, like the one presented in this paper, that has inherent support for non-projective trees. Examples of this include the Chu-Liu-Edmonds' approach for maximum spanning tree (MST) parsing (McDonald et al., 2005) and Nivre's (2009) swap-based reordering method for shift-reduce parsing. The second approach is to create an initial projective parse and then apply transformations to introduce non-projectivity into it. Examples of this include McDonald and Pereira's (2006) rewriting of projective trees produced by the Eisner (1996) algorithm, and Nivre and Nilsson's (2005) pseudo-projective approach that creates projective trees with specially marked arcs that are later transformed into non-projective dependencies.

Descriptive dependency labels. While most recent dependency parsing research has used either vague labels, such as those of the CoNLL shared tasks, or no labels at all, some descriptive dependency label schemes exist. By far the most prominent of these is the Stanford typed dependency scheme (de Marneffe and Manning, 2008). Another descriptive scheme that exists, but which is less widely used in the NLP community, is the one used by Tapanainen and Järvinen's parser (1997). Unfortunately, the Stanford dependency conversion of the Penn Treebank has proven difficult to learn for current dependency parsers (Cer et al., 2010), and there is no publicly available dependency conversion according to Tapanainen and Järvinen's scheme.

Faster parsing. While the fastest reasonable parsing algorithms are the $O(n)$ shift-reduce algorithms, such as Nivre's (2003) algorithm and an expected linear time dynamic programming approach presented by Huang and Sagae (2010), a few other fast alternatives exist. Goldberg and Elhadad's (2010) easy-first algorithm is one such example. Another example, is Roark and Hollingshead's (2009) work that uses chart constraints to achieve linear time complexity for constituency parsing.

Effective features for parsing. A variety of work has investigated the use of more informative features for parsing. This includes work that integrates second and even third order features (McDonald et al., 2006; Carreras, 2007; Koo and Collins, 2010). Also, some work has incorporated unsupervised word clusters as features, including that of Koo et al. (2008) and Suzuki et al. (2009), who utilized unsupervised word clusters created using the Brown et al. (1992) hierarchical clustering algorithm.

Semantically-enriched output. The 2008 and 2009 CoNLL shared tasks (Surdeanu et al., 2008; Hajič et al., 2009), which required participants to build systems capable of both syntactic parsing and Semantic Role Labeling (SRL) (Gildea and Jurafsky, 2002), are the most notable attempts to encour-

[11] These accuracy figures are higher than what should be expected for unseen datasets; see Tratz and Hovy (2010) for more detail.

age the development of parsers with additional semantic annotation. These tasks relied upon PropBank (2005) and NomBank (2004) for the semantic roles. A variety of other systems have focused on FrameNet-based (1998) SRL instead, including those that participated in the SemEval-2007 Task 19 (Baker et al., 2007) and work by Das et al. (2010).

7 Conclusion

In this paper, we have described a new high-quality dependency tree conversion of the Penn Treebank (Marcus, et al., 1993) along with its labeled dependency scheme and presented a parser that is fast, accurate, supports non-projective trees and provides rich output, including not only informative dependency labels similar to Stanford's but also additional semantic annotation for prepositions, possessives, and noun compound relations. We showed how the easy-first algorithm of Goldberg and Elhadad (Goldberg and Elhadad, 2010) can be extended to support non-projective trees by adding *move* actions similar to Nivre's (2009) swap-based reordering for shift-reduce parsing and evaluated our parser on the standard test section of the Penn Treebank, comparing with several other freely available parsers.

The Penn Treebank conversion process fixes a number of buggy trees and part-of-speech tags and produces dependency trees with a relatively small percentage of generic *dep* dependencies. The experimental results show that dependency parsers can generally produce Stanford-granularity labels with high accuracy when using the new dependency conversion of the Penn Treebank, something which, according to the findings of Cer et al. (2010), does not appear to be the case when training and testing dependency parsers on the Stanford conversion.

The parser achieves high labeled and unlabeled accuracy in the evaluation, 92.1% and 93.7%, respectively. The 93.7% result represents a 2.5% increase over the accuracy of Goldberg and Elhadad's (2010) implementation. Also, the parser proves to be quite fast, processing section 23 of the Penn Treebank in just over 30 seconds (a rate of over 75 sentences per second).

The parsing system is capable of not only producing fine-grained dependency relations, but can also produce shallow semantic annotations for prepositions, possessives, and noun compounds by using several optional integrated modules. The preposition sense disambiguation (PSD) module achieves 85.7% accuracy on the SemEval-2007 PSD task, exceeding the previous best published result of 84.8% by a statistically significant margin, the possessives module is over 85% accurate, the noun compound interpretation module achieves 79.6% accuracy on Tratz and Hovy's (2010) dataset. The PropBank SRL module achieves 89.5 F_1 on predicate disambiguation and 85.6 F_1 on argument and adjuncts corresponding to dependency links, for an overall F_1 of 86.8. Combined with the core parser, these modules allow the system to produce a substantially more informative textual analysis than a standard parser.

8 Future Work

There are a variety of ways to extend and improve upon this work. We would like to change our handling of coordinating conjunctions to treat the coordinating conjunction as the head because this has fewer ambiguities than the current approach and also add the ability to produce traces for WH- words. It would also be interesting to examine the impact on final parsing accuracy of the various differences between our dependency conversion and Stanford's.

To aid future NLP research work, the code, including the treebank converter, part-of-speech tagger, parser, and semantic annotation add-ons, will be made publicly available for download via http://www.isi.edu.

Acknowledgements

We would like to thank Richard Johansson for providing us with the code for the *pennconverter* consituent-to-dependency converter. We would also like to thank Dirk Hovy and Anselmo Peñas for many valuable dicussions and suggestions.

References

Collin Baker, and Michael Ellsworth and Katrin Erk. 2007. SemEval'07 task 19: Frame Semantic Structure Extraction. In *Proc. of the 4th International Workshop on Semantic Evaluations*

Collin Baker, Charles J. Fillmore and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proc. of the*

17th international conference on Computational linguistics

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. In *Computational Linguistics* 22(1):39–71

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics* 18(4):467–479.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL 2006*.

Xavier Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.

Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. In *Proc. of LREC 2010*.

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proc. of NAACL 2000*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-find-grained *n*-best parsing and discriminative reranking. In *Proc. of ACL 2005*.

Michael A. Covington. 2001. A Fundamental Algorithm for Dependency Parsing. In *Proc. of the 39th Annual ACM Southeast Conference*.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic Frame-Semantic Parsing. In *Proc. of HLT-NAACL 2010*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *COLING Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Jason Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proc. of COLING 1996*.

Christiane Fellbaum. 1998. WordNet: An Electronic Lexical Database. MIT Press.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. Computational Linguistics. 28(3):245–288.

Jesús Giménez and Lluís Márquez 2004. SVMTool: A General POS Tagger Generator Based on Support Vector Machines. In *Proc. of LREC 2004*.

Yoav Goldberg and Michael Elhadad. 2010. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*.

Yoav Goldberg and Michael Elhadad. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proc. of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*.

Dirk Hovy, Stephen Tratz, and Eduard Hovy. 2010. What's in a Preposition?—Dimensions of Sense Disambiguation for an Interesting Word Class. In *Proc. of COLING 2010*.

Liang Huang and Kenji Sagae. 2010. Dynamic Programming for Linear-Time Shift-Reduce Parsing. In *Proc. of ACL 2010*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *Proc. of NODALIDA*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proc. of ACL 2008*.

Terry Koo and Michael Collins. 2010. Efficient Third-order Dependency Parsers. In *Proc. of ACL 2010*.

Ken Litkowski and Orin Hargraves. 2007. SemEval-2007 Task 06: Word-Sense Disambiguation of Prepositions. In *Proc. of the 4th International Workshop on Semantic Evaluations*.

Christopher D. Manning. 2011. Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In *Proc. of the 12th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2011)*.

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

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-Projective Dependency Parsing Using Spanning Tree Algorithms. In *Proc. of HLT-EMNLP 2005*.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proc. of EACL 2006*.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young and Ralph Grishman. 2004. The NomBank Project: An Interim Report. In *Proc. of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*.

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proc. of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*.

Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *Proc. of the 8th International Workshop on Parsing Technologies (IWPT)*.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An Improved Oracle for Dependency Parsing with Online Reordering. In *Proc. of the 11th International Conference on Parsing Technologies (IWPT)*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of EMNLP-CoNLL 2007*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proc. of LREC 2006*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL-2005*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. In *Computational Linguistics*. 31(1):71–106.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proc. of HLT-NAACL 2007*.

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

Brian Roark and Kristy Hollingshead. 2009. Linear complexity context-free parsing pipelines via chart constraints. In *Proc. of HLT-NAACL*.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. Linguistic Data Consortium, Philadelphia.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of the Twelfth Conference on Computational Natural Language Learning*.

Jun Suzuki, Hideki Isozaki, Xavier Carrerras, and Michael Collins. 2009. An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing. In *Proc. of EMNLP*.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proc. of the fifth conference on applied natural language processing*.

Stephen Tratz and Dirk Hovy. 2009. Disambiguation of Preposition Sense using Linguistically Motivated Features. In *Proc. of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*.

Stephen Tratz and Eduard Hovy. 2010. A Taxonomy, Dataset, and Classifier for Automatic Noun Compound Interpretation. In *Proc. of ACL 2010*.

David Vadas and James R. Curran. 2007. Adding Noun Phrase Structure to the Penn Treebank. In *Proc. of ACL 2007*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis With Support Vector Machines. In *Proc. of 8th International Workshop on Parsing Technologies (IWPT)*.

Lateen EM: Unsupervised Training with Multiple Objectives, Applied to Dependency Grammar Induction

Valentin I. Spitkovsky
Computer Science Department
Stanford University and Google Inc.
valentin@cs.stanford.edu

Hiyan Alshawi
Google Inc.
Mountain View, CA, 94043, USA
hiyan@google.com

Daniel Jurafsky
Departments of Linguistics and Computer Science
Stanford University, Stanford, CA, 94305, USA
jurafsky@stanford.edu

Abstract

We present new training methods that aim to mitigate local optima and slow convergence in unsupervised training by using additional imperfect objectives. In its simplest form, *lateen EM* alternates between the two objectives of ordinary "soft" and "hard" expectation maximization (EM) algorithms. Switching objectives when stuck can help escape local optima. We find that applying a single such alternation already yields state-of-the-art results for English dependency grammar induction. More elaborate lateen strategies track *both* objectives, with each validating the moves proposed by the other. Disagreements can signal earlier opportunities to switch or terminate, saving iterations. De-emphasizing fixed points in these ways eliminates some guesswork from tuning EM. An evaluation against a suite of unsupervised dependency parsing tasks, for a variety of languages, showed that lateen strategies significantly speed up training of both EM algorithms, and improve accuracy for hard EM.

1 Introduction

Expectation maximization (EM) algorithms (Dempster et al., 1977) play important roles in learning latent linguistic structure. Unsupervised techniques from this family excel at core natural language processing (NLP) tasks, including segmentation, alignment, tagging and parsing. Typical implementations specify a probabilistic framework, pick an initial model instance, and iteratively improve parameters using EM. A key guarantee is that subsequent model instances are no worse than the previous, according to training data likelihood in the given framework.

Another attractive feature that helped make EM instrumental (Meng, 2007) is its initial efficiency: Training tends to begin with large steps in a parameter space, sometimes bypassing many local optima at once. After a modest number of such iterations, however, EM lands close to an attractor. Next, its convergence rate necessarily suffers: Disproportionately many (and ever-smaller) steps are needed to finally approach this fixed point, which is almost invariably a local optimum. Deciding when to terminate EM often involves guesswork; and finding ways out of local optima requires trial and error. We propose several strategies that address both limitations.

Unsupervised objectives are, at best, loosely correlated with extrinsic performance (Pereira and Schabes, 1992; Merialdo, 1994; Liang and Klein, 2008, *inter alia*). This fact justifies (occasionally) deviating from a prescribed training course. For example, since *multiple* equi-plausible objectives are usually available, a learner could cycle through them, optimizing alternatives when the primary objective function gets stuck; or, instead of trying to escape, it could aim to avoid local optima in the first place, by halting search early if an improvement to one objective would come at the expense of harming another.

We test these general ideas by focusing on non-convex likelihood optimization using EM. This setting is standard and has natural and well-understood objectives: the classic, "soft" EM; and Viterbi, or "hard" EM (Kearns et al., 1997). The name "lateen" comes from the sea — triangular *lateen* sails can take wind on either side, enabling sailing vessels to *tack* (see Figure 1). As a captain can't count on favorable winds, so an unsupervised learner can't rely on co-operative gradients: soft EM maximizes

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1269–1280,
Edinburgh, Scotland, UK, July 27–31, 2011.

Figure 1: A triangular sail atop a traditional Arab sailing vessel, the *dhow* (right). Older square sails permitted sailing only before the wind. But the efficient *lateen* sail worked like a wing (with high pressure on one side and low pressure on the other), allowing a ship to go almost directly into a headwind. By *tacking*, in a zig-zag pattern, it became possible to sail in any direction, provided there was some wind at all (left). For centuries seafarers expertly combined both sails to traverse extensive distances, greatly increasing the reach of medieval navigation.[1]

likelihoods of observed data across assignments to hidden variables, whereas hard EM focuses on most likely completions.[2] These objectives are plausible, yet both can be provably "wrong" (Spitkovsky et al., 2010a, §7.3). Thus, it is permissible for lateen EM to maneuver between their gradients, for example by tacking around local attractors, in a zig-zag fashion.

2 The Lateen Family of Algorithms

We propose several strategies that use a secondary objective to improve over standard EM training. For hard EM, the secondary objective is that of soft EM; and vice versa if soft EM is the primary algorithm.

2.1 Algorithm #1: Simple Lateen EM

Simple lateen EM begins by running standard EM to convergence, using a user-supplied initial model, primary objective and definition of convergence. Next, the algorithm alternates. A single lateen alternation involves two phases: (i) retraining using the secondary objective, starting from the previous converged solution (once again iterating until convergence, but now of the secondary objective); and (ii) retraining using the primary objective again, starting from the latest converged solution (once more to convergence of the primary objective). The algorithm stops upon failing to sufficiently improve the primary objective across alternations (applying the standard convergence criterion end-to-end) and returns the best of all models re-estimated during training (as judged by the primary objective).

2.2 Algorithm #2: Shallow Lateen EM

Same as algorithm #1, but switches back to optimizing the primary objective after a *single* step with the secondary, during phase (i) of all lateen alternations. Thus, the algorithm alternates between optimizing a primary objective to convergence, then stepping away, using one iteration of the secondary optimizer.

2.3 Algorithm #3: Early-Stopping Lateen EM

This variant runs standard EM but quits early if the secondary objective suffers. We redefine convergence by "or"-ing the user-supplied termination criterion (i.e., a "small-enough" change in the primary objective) with *any* adverse change of the secondary (i.e., an increase in its cross-entropy). Early-stopping lateen EM does *not* alternate objectives.

2.4 Algorithm #4: Early-Switching Lateen EM

Same as algorithm #1, but with the new definition of convergence, as in algorithm #3. Early-switching lateen EM halts primary optimizers as soon as they hurt the secondary objective and stops secondary optimizers once they harm the primary objective. This algorithm terminates when it fails to sufficiently improve the primary objective across a full alternation.

2.5 Algorithm #5: Partly-Switching Lateen EM

Same as algorithm #4, but again iterating primary objectives to convergence, as in algorithm #1; secondary optimizers still continue to terminate early.

3 The Task and Study #1

We chose to test the impact of these five lateen algorithms on unsupervised dependency parsing — a task in which EM plays an important role (Paskin, 2001; Klein and Manning, 2004; Gillenwater et al., 2010, *inter alia*). This entailed two sets of experiments: In study #1, we tested whether single alternations of simple lateen EM (as defined in §2.1,

[1] Partially adapted from `http://www.britannica.com/EBchecked/topic/331395`, `http://allitera.tive.org/archives/004922.html` and `http://landscapedvd.com/desktops/images/ship1280x1024.jpg`.

[2] See Brown et al.'s (1993, §6.2) definition of *Viterbi training* for a succinct justification of hard EM; in our case, the corresponding objective is Spitkovsky et al.'s (2010a, §7.1) $\hat{\theta}_{\text{VIT}}$.

System	*DDA (%)*
(Blunsom and Cohn, 2010)	**55.7**
(Gillenwater et al., 2010)	53.3
(Spitkovsky et al., 2010b)	50.4
+ soft EM + hard EM	52.8 (+**2.4**)
lexicalized, using hard EM	54.3 (+1.5)
+ soft EM + hard EM	**55.6** (+**1.3**)

Table 1: Directed dependency accuracies (DDA) on Section 23 of WSJ (all sentences) for recent state-of-the-art systems and our two experiments (one unlexicalized and one lexicalized) with a single alternation of lateen EM.

Algorithm #1) improve our recent publicly-available system for English dependency grammar induction. In study #2, we introduced a more sophisticated methodology that uses factorial designs and regressions to evaluate lateen strategies with unsupervised dependency parsing in many languages, after also controlling for other important sources of variation.

For study #1, our base system (Spitkovsky et al., 2010b) is an instance of the popular (unlexicalized) Dependency Model with Valence (Klein and Manning, 2004). This model was trained using hard EM on WSJ45 (WSJ sentences up to length 45) until successive changes in per-token cross-entropy fell below 2^{-20} bits (Spitkovsky et al., 2010b; 2010a, §4).[3]

We confirmed that the base model had indeed converged, by running 10 steps of hard EM on WSJ45 and verifying that its objective did not change much. Next, we applied a single alternation of simple lateen EM: first running soft EM (this took 101 steps, using the same termination criterion), followed by hard EM (again to convergence — another 23 iterations). The result was a decrease in hard EM's cross-entropy, from 3.69 to 3.59 bits per token (bpt), accompanied by a 2.4% jump in accuracy, from 50.4 to 52.8%, on Section 23 of WSJ (see Table 1).[4]

Our first experiment showed that lateen EM holds promise for simple models. Next, we tested it in a more realistic setting, by re-estimating *lexicalized* models,[5] starting from the unlexicalized model's parses; this took 24 steps with hard EM. We then applied another single lateen alternation: This time, soft EM ran for 37 steps, hard EM took another 14, and the new model again improved, by 1.3%, from 54.3 to 55.6% (see Table 1); the corresponding drop in (lexicalized) cross-entropy was from 6.10 to 6.09 bpt. This last model is competitive with the state-of-the-art; moreover, gains from single applications of simple lateen alternations (2.4 and 1.3%) are on par with the increase due to lexicalization alone (1.5%).

4 Methodology for Study #2

Study #1 suggests that lateen EM can improve grammar induction in English. To establish statistical significance, however, it is important to test a hypothesis in many settings (Ioannidis, 2005). We therefore use a factorial experimental design and regression analyses with a variety of lateen strategies. Two regressions — one predicting accuracy, the other, the number of iterations — capture the effects that lateen algorithms have on performance and efficiency, relative to standard EM training. We controlled for important dimensions of variation, such as the underlying language: to make sure that our results are not English-specific, we induced grammars in 19 languages. We also explored the impact from the quality of an initial model (using both uniform and ad hoc initializers), the choice of a primary objective (i.e., soft or hard EM), and the quantity and complexity of training data (shorter versus both short and long sentences). Appendix A gives the full details.

4.1 Data Sets

We use all 23 train/test splits from the 2006/7 CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007),[6] which cover 19 different languages.[7] We splice out all punctuation labeled in the data, as is standard practice (Paskin, 2001; Klein and Manning, 2004), introducing new arcs from grandmothers to grand-daughters where necessary, both in train- and test-sets. Evaluation is always against the

[3] `http://nlp.stanford.edu/pubs/markup-data.tar.bz2: dp.model.dmv`

[4] It is standard practice to convert gold labeled constituents from Penn English Treebank's Wall Street Journal (WSJ) portion (Marcus et al., 1993) into unlabeled reference dependency parses using deterministic "head-percolation" rules (Collins, 1999); sentence root symbols (but not punctuation) arcs count towards accuracies (Paskin, 2001; Klein and Manning, 2004).

[5] We used Headden et al.'s (2009) method (also the approach taken by the two stronger state-of-the-art systems): for words seen at least 100 times in the training corpus, gold part-of-speech tags are augmented with lexical items.

[6] These disjoint splits require smoothing; in the WSJ setting, training and test sets overlapped (Klein and Manning, 2004).

[7] We down-weigh languages appearing in both years — Arabic, Chinese, Czech and Turkish — by 50% in all our analyses.

entire resulting test sets (i.e., all sentence lengths).[8]

4.2 Grammar Models

In all remaining experiments we model grammars via the original DMV, which ignores punctuation; all models are unlexicalized, with gold part-of-speech tags for word classes (Klein and Manning, 2004).

4.3 Smoothing Mechanism

All unsmoothed models are smoothed immediately prior to evaluation; some of the baseline models are also smoothed during training. In both cases, we use the "add-one" (a.k.a. Laplace) smoothing algorithm.

4.4 Standard Convergence

We always halt an optimizer once a change in its objective's consecutive cross-entropy values falls below 2^{-20} bpt (at which point we consider it "stuck").

4.5 Scoring Function

We report directed accuracies — fractions of correctly guessed (unlabeled) dependency arcs, including arcs from sentence root symbols, as is standard practice (Paskin, 2001; Klein and Manning, 2004). Punctuation does not affect scoring, as it had been removed from all parse trees in our data (see §4.1).

5 Experiments

We now summarize our baseline models and briefly review the proposed lateen algorithms. For details of the default systems (standard soft and hard EM), all control variables and both regressions (against final accuracies and iteration counts) see Appendix A.

5.1 Baseline Models

We tested a total of six baseline models, experimenting with two types of alternatives: (i) strategies that perturb stuck models directly, by *smoothing*, ignoring secondary objectives; and (ii) *shallow* applications of a single EM step, ignoring convergence.

Baseline $B1$ alternates running standard EM to convergence and smoothing. A second baseline, $B2$, smooths after every step of EM instead. Another shallow baseline, $B3$, alternates single steps of soft and hard EM.[9] Three such baselines begin with hard EM (marked with the subscript h); and three more start with soft EM (marked with the subscript s).

5.2 Lateen Models

Ten models, $A\{1,2,3,4,5\}_{\{h,s\}}$, correspond to our lateen algorithms #1–5 (§2), starting with either hard or soft EM's objective, to be used as the primary.

6 Results

		Soft EM		Hard EM	
	Model	Δa	Δi	Δa	Δi
Baselines	$B3$	**-2.7**	$\times$**0.2**	**-2.0**	$\times$**0.3**
	$B2$	+0.6	$\times$**0.7**	+0.6	$\times$1.2
	$B1$	0.0	$\times$**2.0**	+0.8	$\times$**3.7**
Algorithms	$A1$	0.0	$\times$**1.3**	**+5.5**	$\times$**6.5**
	$A2$	-0.0	$\times$**1.3**	+1.5	$\times$**3.6**
	$A3$	0.0	$\times$**0.7**	-0.1	$\times$**0.7**
	$A4$	0.0	$\times$**0.8**	**+3.0**	$\times$**2.1**
	$A5$	0.0	$\times$**1.2**	**+2.9**	$\times$**3.8**

Table 2: Estimated additive changes in directed dependency accuracy (Δa) and multiplicative changes in the number of iterations before terminating (Δi) for all baseline models and lateen algorithms, relative to standard training: soft EM (left) and hard EM (right). Bold entries are statistically different ($p < 0.01$) from zero, for Δa, and one, for Δi (details in Table 4 and Appendix A).

Not one baseline attained a statistically significant performance improvement. Shallow models $B3_{\{h,s\}}$, in fact, significantly lowered accuracy: by 2.0%, on average ($p \approx 7.8 \times 10^{-4}$), for $B3_h$, which began with hard EM; and down 2.7% on average ($p \approx 6.4 \times 10^{-7}$), for $B3_s$, started with soft EM. They were, however, 3–5x faster than standard training, on average (see Table 4 for all estimates and associated p-values; above, Table 2 shows a preview of the full results).

6.1 $A1_{\{h,s\}}$ — Simple Lateen EM

$A1_h$ runs 6.5x slower, but scores 5.5% higher, on average, compared to standard Viterbi training; $A1_s$ is only 30% slower than standard soft EM, but does not impact its accuracy at all, on average.

Figure 2 depicts a sample training run: Italian '07 with $A1_h$. Viterbi EM converges after 47 iterations,

[8] With the exception of Arabic '07, from which we discarded a single sentence containing 145 non-punctuation tokens.

[9] It approximates a mixture (the average of soft and hard objectives) — a natural comparison, computable via gradients and standard optimization algorithms, such as L-BFGS (Liu and Nocedal, 1989). We did not explore exact interpolations, however, because replacing EM is itself a significant confounder, even with unchanged objectives (Berg-Kirkpatrick et al., 2010).

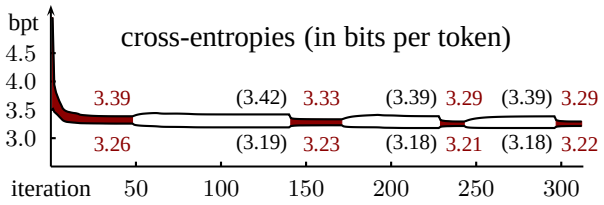


Figure 2: Cross-entropies for Italian '07, initialized uniformly and trained on sentences up to length 45. The two curves are primary and secondary objectives (soft EM's lies below, as sentence yields are at least as likely as parse trees): shaded regions indicate iterations of hard EM (primary); and annotated values are measurements upon each optimizer's convergence (soft EM's are parenthesized).

reducing the primary objective to 3.39 bpt (the secondary is then at 3.26); accuracy on the held-out set is 41.8%. Three alternations of lateen EM (totaling 265 iterations) further decrease the primary objective to 3.29 bpt (the secondary also declines, to 3.22) and accuracy increases to 56.2% (14.4% higher).

6.2 $A2_{\{h,s\}}$ — Shallow Lateen EM

$A2_h$ runs 3.6x slower, but scores only 1.5% higher, on average, compared to standard Viterbi training; $A2_s$ is again 30% slower than standard soft EM and also has no measurable impact on parsing accuracy.

6.3 $A3_{\{h,s\}}$ — Early-Stopping Lateen EM

Both $A3_h$ and $A3_s$ run 30% faster, on average, than standard training with hard or soft EM; and neither heuristic causes a statistical change to accuracy.

Table 3 shows accuracies and iteration counts for 10 (of 23) train/test splits that terminate early with $A3_s$ (in one particular, example setting). These runs are nearly twice as fast, and only two score (slightly) lower, compared to standard training using soft EM.

6.4 $A4_{\{h,s\}}$ — Early-Switching Lateen EM

$A4_h$ runs only 2.1x slower, but scores only 3.0% higher, on average, compared to standard Viterbi training; $A4_s$ is, in fact, 20% faster than standard soft EM, but still has no measurable impact on accuracy.

6.5 $A5_{\{h,s\}}$ — Partly-Switching Lateen EM

$A5_h$ runs 3.8x slower, scoring 2.9% higher, on average, compared to standard Viterbi training; $A5_s$ is 20% slower than soft EM, but, again, no more accurate. Indeed, $A4$ strictly dominates both $A5$ variants.

CoNLL Year & Language	Soft EM		$A3_s$	
	DDA	*iters*	*DDA*	*iters*
Arabic 2006	28.4	180	28.4	118
Bulgarian '06	39.1	253	**39.6**	131
Chinese '06	49.4	268	49.4	204
Dutch '06	21.3	246	**27.8**	35
Hungarian '07	17.1	366	**17.4**	213
Italian '07	39.6	194	39.6	164
Japanese '06	56.6	113	56.6	93
Portuguese '06	37.9	180	*37.5*	102
Slovenian '06	30.8	234	**31.1**	118
Spanish '06	33.3	125	*33.1*	73
Average:	35.4	216	36.1	125

Table 3: Directed dependency accuracies (DDA) and iteration counts for the 10 (of 23) train/test splits affected by early termination (setting: soft EM's primary objective, trained using shorter sentences and ad-hoc initialization).

7 Discussion

Lateen strategies improve dependency grammar induction in several ways. Early stopping offers a clear benefit: 30% higher efficiency yet same performance as standard training. This technique could be used to (more) fairly compare learners with radically different objectives (e.g., lexicalized and unlexicalized), requiring quite different numbers of steps (or magnitude changes in cross-entropy) to converge.

The second benefit is improved performance, but only starting with hard EM. Initial local optima discovered by soft EM are such that the impact on accuracy of all subsequent heuristics is indistinguishable from noise (it's not even negative). But for hard EM, lateen strategies consistently improve accuracy — by 1.5, 3.0 or 5.5% — as an algorithm follows the secondary objective longer (a single step, until the primary objective gets worse, or to convergence).

Our results suggest that soft EM should use early termination to improve efficiency. Hard EM, by contrast, could use any lateen strategy to improve either efficiency or performance, or to strike a balance.

8 Related Work

8.1 Avoiding and/or Escaping Local Attractors

Simple lateen EM is similar to Dhillon et al.'s (2002) refinement algorithm for text clustering with spherical k-means. Their "ping-pong" strategy alternates batch and incremental EM, exploits the strong points of each, and improves a *shared* objective at every

step. Unlike generalized (GEM) variants (Neal and Hinton, 1999), lateen EM uses multiple objectives: it sacrifices the primary in the short run, to escape local optima; in the long run, it also does no harm, by construction (as it returns the best model seen).

Of the meta-heuristics that use more than a standard, scalar objective, deterministic annealing (DA) (Rose, 1998) is closest to lateen EM. DA perturbs objective functions, instead of manipulating solutions directly. As other continuation methods (Allgower and Georg, 1990), it optimizes an easy (e.g., convex) function first, then "rides" that optimum by gradually morphing functions towards the difficult objective; each step reoptimizes from the previous approximate solution. Smith and Eisner (2004) employed DA to improve part-of-speech disambiguation, but found that objectives had to be further "skewed," using domain knowledge, before it helped (constituent) grammar induction. (For this reason, we did not experiment with DA, despite its strong similarities to lateen EM.) Smith and Eisner (2004) used a "temperature" β to anneal a flat uniform distribution ($\beta = 0$) into soft EM's non-convex objective ($\beta = 1$). In their framework, hard EM corresponds to $\beta \longrightarrow \infty$, so the algorithms differ only in their β-schedule: DA's is continuous, from 0 to 1; lateen EM's is a discrete alternation, of 1 and $+\infty$.[10]

8.2 Terminating Early, Before Convergence

EM is rarely run to (even numerical) convergence. Fixing a modest number of iterations a priori (Klein, 2005, §5.3.4), running until successive likelihood ratios become small (Spitkovsky et al., 2009, §4.1) or using a combination of the two (Ravi and Knight, 2009, §4, Footnote 5) is standard practice in NLP. Elworthy's (1994, §5, Figure 1) analysis of part-of-speech tagging showed that, in most cases, a small number of iterations is actually preferable to convergence, in terms of final accuracies: "regularization by early termination" had been suggested for image deblurring algorithms in statistical astronomy (Lucy, 1974, §2); and validation against held-out data — a strategy proposed much earlier, in psychology (Larson, 1931), has also been used as a halting criterion in NLP (Yessenalina et al., 2010, §4.2, 5.2). Early-stopping lateen EM tethers termination to a *sign* change in the direction of a secondary objective, similarly to (cross-)validation (Stone, 1974; Geisser, 1975; Arlot and Celisse, 2010), but without splitting data — it trains using all examples, at all times.[11,12]

8.3 Training with Multiple Views

Lateen strategies may seem conceptually related to co-training (Blum and Mitchell, 1998). However, bootstrapping methods generally begin with some labeled data and gradually label the rest (discriminatively) as they grow more confident, but do not optimize an explicit objective function; EM, on the other hand, can be fully unsupervised, relabels all examples on each iteration (generatively), and guarantees not to hurt a well-defined objective, at every step.[13] Co-training classically relies on two views of the data — redundant feature sets that allow different algorithms to label examples for each other, yielding "probably approximately correct" (PAC)-style guarantees under certain (strong) assumptions. In contrast, lateen EM uses the same data, features, model and essentially the same algorithms, changing only their objective functions: it makes no assumptions, but guarantees not to harm the primary objective.

Some of these distinctions have become blurred with time: Collins and Singer (1999) introduced an objective function (also based on agreement) into co-training; Goldman and Zhou (2000), Ng and Cardie (2003) and Chan et al. (2004) made do without redundant views; Balcan et al. (2004) relaxed other strong assumptions; and Zhou and Goldman (2004) generalized co-training to accommodate three and more algorithms. Several such methods have been applied to dependency parsing (Søgaard and Rishøj, 2010), constituent parsing (Sarkar,

[10] One can think of this as a kind of "beam search" (Lowerre, 1976), with soft EM expanding and hard EM pruning a frontier.

[11] We see in it a milder contrastive estimation (Smith and Eisner, 2005a; 2005b), agnostic to implicit negative evidence, but caring *whence* learners push probability mass towards training examples: when most likely parse trees begin to benefit at the expense of their sentence yields (or vice versa), optimizers halt.

[12] For a recently proposed instance of EM that uses cross-validation (CV) to optimize *smoothed* data likelihoods (in learning synchronous PCFGs, for phrase-based machine translation), see Mylonakis and Sima'an's (2010, §3.1) CV-EM algorithm.

[13] Some authors (Nigam and Ghani, 2000; Ng and Cardie, 2003; Smith and Eisner, 2005a, §5.2, 7; §2; §6) draw a hard line between bootstrapping algorithms, such as self- and co-training, and probabilistic modeling using EM; others (Dasgupta et al., 2001; Chang et al., 2007, §1; §5) tend to lump them together.

2001) and parser reranking (Crim, 2002). Fundamentally, co-training exploits redundancies in unlabeled data and/or learning algorithms. Lateen strategies *also* exploit redundancies: in noisy objectives. Both approaches use a second vantage point to improve their perception of difficult training terrains.

9 Conclusions and Future Work

Lateen strategies can improve performance and efficiency for dependency grammar induction with the DMV. Early-stopping lateen EM is 30% faster than standard training, without affecting accuracy — it reduces guesswork in terminating EM. At the other extreme, simple lateen EM is slower, but significantly improves accuracy — by 5.5%, on average — for hard EM, escaping some of its local optima.

It would be interesting to apply lateen algorithms to advanced parsing models (Blunsom and Cohn, 2010; Headden et al., 2009, *inter alia*) and learning algorithms (Gillenwater et al., 2010; Cohen and Smith, 2009, *inter alia*). Future work could explore other NLP tasks — such as clustering, sequence labeling, segmentation and alignment — that often employ EM. Our meta-heuristics are multi-faceted, featuring aspects of iterated local search, deterministic annealing, cross-validation, contrastive estimation and co-training. They may be generally useful in machine learning and non-convex optimization.

Appendix A. Experimental Design

Statistical techniques are vital to many aspects of computational linguistics (Johnson, 2009; Charniak, 1997; Abney, 1996, *inter alia*). We used factorial designs,[14] which are standard throughout the natural and social sciences, to assist with experimental design and statistical analyses. Combined with ordinary regressions, these methods provide succinct and interpretable summaries that explain which settings meaningfully contribute to changes in dependent variables, such as running time and accuracy.

[14] We used *full* factorial designs for clarity of exposition. But many fewer experiments would suffice, especially in regression models without interaction terms: for the more efficient *fractional* factorial designs, as well as for randomized block designs and full factorial designs, see Montgomery (2005, Ch. 4–9).

9.1 Dependent Variables

We constructed two regressions, for two types of dependent variables: to summarize performance, we predict accuracies; and to summarize efficiency, we predict (logarithms of) iterations before termination.

In the performance regression, we used four different scores for the dependent variable. These include both directed accuracies and *undirected* accuracies, each computed in two ways: (i) using a best parse tree; and (ii) using all parse trees. These four types of scores provide different kinds of information. Undirected scores ignore polarity of parent-child relations (Paskin, 2001; Klein and Manning, 2004; Schwartz et al., 2011), partially correcting for some effects of alternate analyses (e.g., systematic choices between modals and main verbs for heads of sentences, determiners for noun phrases, etc.). And *integrated* scoring, using the inside-outside algorithm (Baker, 1979) to compute expected accuracy across all — not just best — parse trees, has the advantage of incorporating probabilities assigned to individual arcs: This metric is more sensitive to the margins that separate best from next-best parse trees, and is not affected by tie-breaking. We tag scores using two binary predictors in a simple (first order, multi-linear) regression, where having multiple relevant quality assessments improves goodness-of-fit.

In the efficiency regression, dependent variables are logarithms of the numbers of iterations. Wrapping EM in an inner loop of a heuristic has a multiplicative effect on the total number of models re-estimated prior to termination. Consequently, logarithms of the final counts better fit the observed data.

9.2 Independent Predictors

All of our predictors are binary indicators (a.k.a. "dummy" variables). The *undirected* and *integrated* factors only affect the regression for accuracies (see Table 4, left); remaining factors participate also in the running times regression (see Table 4, right). In a default run, all factors are zero, corresponding to the intercept estimated by a regression; other estimates reflect changes in the dependent variable associated with having that factor "on" instead of "off."

- *adhoc* — This setting controls initialization. By default, we use the uninformed uniform initializer (Spitkovsky et al., 2010a); when it is

	Goodness-of-Fit:		Regression for **Accuracies** ($R^2_{adj} \approx 76.2\%$)		Regression for ln(**Iterations**) ($R^2_{adj} \approx 82.4\%$)		
	Indicator Factors		*coeff.* $\hat{\beta}$	*adj. p-value*			
	undirected		**18.1**	$< 2.0 \times 10^{-16}$			
	integrated		**-0.9**	$\approx 7.0 \times 10^{-7}$	*coeff.* $\hat{\beta}$	*mult.* $e^{\hat{\beta}}$	*adj. p-value*
		(intercept)	**30.9**	$< 2.0 \times 10^{-16}$	5.5	**255.8**	$< 2.0 \times 10^{-16}$
	adhoc		**1.2**	$\approx 3.1 \times 10^{-13}$	-0.0	1.0	≈ 1.0
Model	*sweet*		**1.0**	$\approx 3.1 \times 10^{-9}$	-0.2	**0.8**	$< 2.0 \times 10^{-16}$
$B3_s$	*shallow*	*(soft-first)*	**-2.7**	$\approx 6.4 \times 10^{-7}$	-1.5	**0.2**	$< 2.0 \times 10^{-16}$
$B3_h$	*shallow*	*(hard-first)*	**-2.0**	$\approx 7.8 \times 10^{-4}$	-1.2	**0.3**	$< 2.0 \times 10^{-16}$
$B2_s$		*shallow smooth*	0.6	≈ 1.0	-0.4	**0.7**	$\approx 1.4 \times 10^{-12}$
$B1_s$		*smooth*	0.0	≈ 1.0	0.7	**2.0**	$< 2.0 \times 10^{-16}$
$A1_s$		*simple lateen*	0.0	≈ 1.0	0.2	**1.3**	$\approx 4.1 \times 10^{-4}$
$A2_s$		*shallow lateen*	-0.0	≈ 1.0	0.2	**1.3**	$\approx 5.8 \times 10^{-4}$
$A3_s$		*early-stopping lateen*	0.0	≈ 1.0	-0.3	**0.7**	$\approx 2.6 \times 10^{-7}$
$A4_s$		*early-switching lateen*	0.0	≈ 1.0	-0.3	**0.8**	$\approx 2.6 \times 10^{-7}$
$A5_s$		*partly-switching lateen*	0.0	≈ 1.0	0.2	**1.2**	$\approx 4.2 \times 10^{-3}$
	viterbi		**-4.0**	$\approx 5.7 \times 10^{-16}$	-1.7	**0.2**	$< 2.0 \times 10^{-16}$
$B2_h$		*shallow smooth*	0.6	≈ 1.0	0.2	1.2	$\approx 5.6 \times 10^{-2}$
$B1_h$		*smooth*	0.8	≈ 1.0	1.3	**3.7**	$< 2.0 \times 10^{-16}$
$A1_h$		*simple lateen*	**5.5**	$< 2.0 \times 10^{-16}$	1.9	**6.5**	$< 2.0 \times 10^{-16}$
$A2_h$		*shallow lateen*	1.5	$\approx 5.0 \times 10^{-2}$	1.3	**3.6**	$< 2.0 \times 10^{-16}$
$A3_h$		*early-stopping lateen*	-0.1	≈ 1.0	-0.4	**0.7**	$\approx 1.7 \times 10^{-11}$
$A4_h$		*early-switching lateen*	**3.0**	$\approx 1.0 \times 10^{-8}$	0.7	**2.1**	$< 2.0 \times 10^{-16}$
$A5_h$		*partly-switching lateen*	**2.9**	$\approx 7.6 \times 10^{-8}$	1.3	**3.8**	$< 2.0 \times 10^{-16}$

Table 4: Regressions for accuracies and natural-log-iterations, using 86 binary predictors (all p-values jointly adjusted for simultaneous hypothesis testing; {*langyear*} indicators not shown). Accuracies' estimated coefficients $\hat{\beta}$ that are statistically different from 0 — and iteration counts' multipliers $e^{\hat{\beta}}$ significantly different from 1 — are shown in bold.

on, we use Klein and Manning's (2004) "adhoc" harmonic heuristic, bootstrapped using sentences up to length 10, from the training set.

- *sweet* — This setting controls the length cutoff. By default, we train with all sentences containing up to 45 tokens; when it is on, we use Spitkovsky et al.'s (2009) "sweet spot" cutoff of 15 tokens (recommended for English, WSJ).
- *viterbi* — This setting controls the primary objective of the learning algorithm. By default, we run soft EM; when it is on, we use hard EM.
- $\{\textit{langyear}_i\}_{i=1}^{22}$ — This is a set of 22 mutually-exclusive selectors for the language/year of a train/test split; default (all zeros) is English '07.

Due to space limitations, we exclude *langyear* predictors from Table 4. Further, we do not explore (even two-way) interactions between predictors.[15]

9.3 Statistical Significance

Our statistical analyses relied on the R package (R Development Core Team, 2011), which does not, by default, adjust statistical significance (p-values) for multiple hypotheses testing.[16] We corrected this using the Holm-Bonferroni method (Holm, 1979), which is uniformly more powerful than the older (Dunn-)Bonferroni procedure; since we tested many fewer hypotheses ($44 + 42$ — one per intercept/coefficient $\hat{\beta}$) than settings combinations, its adjustments to the p-values are small (see Table 4).[17]

[15] This approach may miss some interesting facts, e.g., that the *adhoc* initializer is exceptionally good for English, with soft EM. Instead it yields coarse summaries of regularities supported by overwhelming evidence across data and training regimes.

[16] Since we would *expect* $p\%$ of randomly chosen hypotheses to appear significant at the $p\%$ level simply by *chance*, we must take precautions against these and other "data-snooping" biases.

[17] We adjusted the p-values for all 86 hypotheses jointly, using `http://rss.acs.unt.edu/Rdoc/library/multtest/html/mt.rawp2adjp.html`.

CoNLL Year & Language		$A3_s$		Soft EM		$A3_h$		Hard EM		$A1_h$	
		DDA	*iters*	*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*
Arabic	2006	28.4	118	28.4	162	21.6	19	21.6	21	**32.1**	200
	'7	–	–	**26.9**	171	24.7	17	24.8	24	22.0	239
Basque	'7	–	–	39.9	180	32.0	16	32.2	20	**43.6**	128
Bulgarian	'6	39.6	131	39.1	253	41.6	22	41.5	25	**44.3**	140
Catalan	'7	–	–	58.5	135	50.1	48	50.1	54	**63.8**	279
Chinese	'6	**49.4**	204	49.4	268	31.3	24	31.6	55	37.9	378
	'7	–	–	**46.0**	262	30.0	25	30.2	64	34.5	307
Czech	'6	–	–	**50.5**	294	27.8	27	27.7	33	35.2	445
	'7	–	–	**49.8**	263	29.0	37	29.0	41	31.4	307
Danish	'6	–	–	43.5	116	43.8	31	43.9	45	**44.0**	289
Dutch	'6	27.8	35	21.3	246	24.9	44	24.9	49	**32.5**	241
English	'7	–	–	**38.1**	180	34.0	32	33.9	42	34.9	186
German	'6	–	–	33.3	136	25.4	20	25.4	39	**33.5**	155
Greek	'7	–	–	17.5	230	18.3	18	18.3	21	**21.4**	117
Hungarian	'7	17.4	213	17.1	366	12.3	26	12.4	36	**23.0**	246
Italian	'7	**39.6**	164	39.6	194	32.6	25	32.6	27	37.6	273
Japanese	'6	**56.6**	93	56.6	113	49.6	20	49.7	23	53.5	91
Portuguese	'6	37.5	102	**37.9**	180	28.6	27	28.9	41	34.4	134
Slovenian	'6	31.1	118	30.8	234	–	–	23.4	22	**33.6**	255
Spanish	'6	33.1	73	**33.3**	125	18.2	29	18.4	36	33.3	235
Swedish	'6	–	–	41.8	242	36.0	24	36.1	29	**42.5**	296
Turkish	'6	–	–	29.8	303	17.8	19	22.2	38	**31.9**	134
	'7	–	–	28.3	227	14.0	9	10.7	31	**33.4**	242
	Average:	**37.4**	162	37.0	206	30.0	26	30.0	35	37.1	221

Table 5: Performance (directed dependency accuracies measured against all sentences in the evaluation sets) and efficiency (numbers of iterations) for standard training (soft and hard EM), early-stopping lateen EM ($A3$) and simple lateen EM with hard EM's primary objective ($A1_h$), for all 23 train/test splits, with *adhoc* and *sweet* settings on.

9.4 Interpretation

Table 4 shows the estimated coefficients and their (adjusted) p-values for both intercepts and most predictors (excluding the language/year of the data sets) for all 1,840 experiments. The default (English) system uses soft EM, trains with both short and long sentences, and starts from an uninformed uniform initializer. It is estimated to score 30.9%, converging after approximately 256 iterations (both intercepts are statistically different from zero: $p < 2.0 \times 10^{-16}$).

As had to be the case, we detect a gain from *undirected* scoring; *integrated* scoring is slightly (but significantly: $p \approx 7.0 \times 10^{-7}$) negative, which is reassuring: best parses are scoring higher than the rest and may be standing out by large margins. The *adhoc* initializer boosts accuracy by 1.2%, overall (also significant: $p \approx 3.1 \times 10^{-13}$), without a measurable impact on running time ($p \approx 1.0$). Training with fewer, shorter sentences, at the *sweet* spot gradation, adds 1.0% and shaves 20% off the total number of iterations, on average (both estimates are significant).

We find the *viterbi* objective harmful — by 4.0%, on average ($p \approx 5.7 \times 10^{-16}$) — for the CoNLL sets. Spitkovsky et al. (2010a) reported that it helps on WSJ, at least with long sentences and uniform initializers. Half of our experiments are with shorter sentences, and half use ad hoc initializers (i.e., three quarters of settings are not ideal for Viterbi EM), which may have contributed to this negative result; still, our estimates do confirm that hard EM is significantly (80%, $p < 2.0 \times 10^{-16}$) faster than soft EM.

9.5 More on Viterbi Training

The overall negative impact of Viterbi objectives is a cause for concern: On average, $A1_h$'s estimated gain of 5.5% should more than offset the expected 4.0% loss from starting with hard EM. But it is, nevertheless, important to make sure that simple lateen EM with hard EM's primary objective is in fact an improvement over *both* standard EM algorithms.

Table 5 shows performance and efficiency numbers for $A1_h$, $A3_{\{h,s\}}$, as well as standard soft and hard EM, using settings that are least favorable for

CoNLL Year & Language		$A3_s$		Soft EM		$A3_h$		Hard EM		$A1_h$	
		DDA	*iters*	*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*
Arabic	2006	–	–	**33.4**	317	20.8	8	20.2	32	16.6	269
	’7	18.6	60	8.7	252	26.5	9	26.4	14	**49.5**	171
Basque	’7	–	–	18.3	245	23.2	16	23.0	23	**24.0**	162
Bulgarian	’6	27.0	242	27.1	293	40.6	33	40.5	34	**43.9**	276
Catalan	’7	15.0	74	13.8	159	53.2	30	53.1	31	**59.8**	176
Chinese	’6	63.5	131	**63.6**	261	36.8	45	36.8	47	44.5	213
	’7	**58.5**	130	58.5	258	35.2	20	35.0	48	43.2	372
Czech	’6	29.5	125	**29.7**	224	23.6	18	23.8	41	27.7	179
	’7	–	–	25.9	215	27.1	37	27.2	64	**28.4**	767
Danish	’6	–	–	16.6	155	28.7	30	28.7	30	**38.3**	241
Dutch	’6	20.4	51	21.2	174	25.5	30	25.6	38	**27.8**	243
English	’7	–	–	18.0	162	–	–	38.7	35	**45.2**	366
German	’6	–	–	24.4	148	30.1	39	30.1	44	**30.4**	185
Greek	’7	**25.5**	133	25.3	156	–	–	13.2	27	13.2	252
Hungarian	’7	–	–	18.9	310	28.9	34	28.9	44	**34.7**	414
Italian	’7	25.4	127	25.3	165	–	–	**52.3**	36	52.3	81
Japanese	’6	–	–	39.3	143	42.2	38	42.4	48	**50.2**	199
Portuguese	’6	35.2	48	35.6	224	–	–	34.5	21	**36.7**	143
Slovenian	’6	24.8	182	25.3	397	28.8	17	28.8	20	**32.2**	121
Spanish	’6	–	–	27.7	252	–	–	28.3	31	**50.6**	130
Swedish	’6	27.9	49	32.6	287	45.2	22	45.6	52	**50.0**	314
Turkish	’6	–	–	30.5	239	30.2	16	**30.6**	24	29.0	138
	’7	–	–	**48.8**	254	34.3	24	33.1	34	35.9	269
	Average:	27.3	161	27.3	225	33.2	28	33.2	35	**38.2**	236

Table 6: Performance (directed dependency accuracies measured against all sentences in the evaluation sets) and efficiency (numbers of iterations) for standard training (soft and hard EM), early-stopping lateen EM ($A3$) and simple lateen EM with hard EM's primary objective ($A1_h$), for all 23 train/test splits, with setting *adhoc* off and *sweet* on.

Viterbi training: *adhoc* and *sweet* on. Although $A1_h$ scores 7.1% higher than hard EM, on average, it is only slightly better than soft EM — up 0.1% (and worse than $A1_s$). Without *adhoc* (i.e., using uniform initializers — see Table 6), however, hard EM still improves, by 3.2%, on average, whereas soft EM drops nearly 10%; here, $A1_h$ further improves over hard EM, scoring 38.2% (up 5.0), higher than soft EM's accuracies from *both* settings (27.3 and 37.0).

This suggests that $A1_h$ is indeed better than both standard EM algorithms. We suspect that our experimental set-up may be disadvantageous for Viterbi training, since half the settings use ad hoc initializers, and because CoNLL sets are small. (Viterbi EM works best with more data and longer sentences.)

Acknowledgments

Partially funded by the Air Force Research Laboratory (AFRL), under prime contract no. FA8750-09-C-0181, and by NSF, via award #IIS-0811974. We thank Angel X. Chang, Spence Green, David McClosky, Fernando Pereira, Slav Petrov and the anonymous reviewers, for many helpful comments on draft versions of this paper, and Andrew Y. Ng, for a stimulating discussion. First author is grateful to Lynda K. Dunnigan for first introducing him to lateen sails, among other connections, in *Humanities*.

References

S. Abney. 1996. Statistical methods and linguistics. In J. L. Klavans and P. Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. MIT Press.

E. L. Allgower and K. Georg. 1990. *Numerical Continuation Methods: An Introduction*. Springer-Verlag.

S. Arlot and A. Celisse. 2010. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*.

M.-F. Balcan, A. Blum, and K. Yang. 2004. Co-training and expansion: Towards bridging theory and practice. In *NIPS*.

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *NAACL-HLT*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *EMNLP*.

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

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

J. Chan, I. Koprinska, and J. Poon. 2004. Co-training with a single natural feature set applied to email classification. In *WI*.

M.-W. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*.

E. Charniak. 1997. Statistical techniques for natural language parsing. *AI Magazine*, 18.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL-HLT*.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *EMNLP*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

J. Crim. 2002. Co-training re-rankers for improved parser accuracy.

S. Dasgupta, M. L. Littman, and D. McAllester. 2001. PAC generalization bounds for co-training. In *NIPS*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39.

I. S. Dhillon, Y. Guan, and J. Kogan. 2002. Iterative clustering of high dimensional text data augmented by local search. In *ICDM*.

D. Elworthy. 1994. Does Baum-Welch re-estimation help taggers? In *ANLP*.

S. Geisser. 1975. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. 2010. Posterior sparsity in unsupervised dependency parsing. Technical report, University of Pennsylvania.

S. Goldman and Y. Zhou. 2000. Enhancing supervised learning with unlabeled data. In *ICML*.

W. P. Headden, III, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *NAACL-HLT*.

S. Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6.

J. P. A. Ioannidis. 2005. Why most published research findings are false. *PLoS Medicine*, 2.

M. Johnson. 2009. How the statistical revolution changes (computational) linguistics. In *EACL: Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*

M. Kearns, Y. Mansour, and A. Y. Ng. 1997. An information-theoretic analysis of hard and soft assignment methods for clustering. In *UAI*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

D. Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

S. C. Larson. 1931. The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, 22.

P. Liang and D. Klein. 2008. Analyzing the errors of unsupervised learning. In *HLT-ACL*.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming. Series B*, 45.

B. T. Lowerre. 1976. *The* HARPY *Speech Recognition System*. Ph.D. thesis, CMU.

L. B. Lucy. 1974. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

X.-L. Meng. 2007. EM and MCMC: Workhorses for scientific computing (thirty years of EM and much more). *Statistica Sinica*, 17.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20.

D. C. Montgomery. 2005. *Design and Analysis of Experiments*. John Wiley & Sons, 6th edition.

M. Mylonakis and K. Sima'an. 2010. Learning probabilistic synchronous CFGs for phrase-based translation. In *CoNLL*.

R. M. Neal and G. E. Hinton. 1999. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press.

V. Ng and C. Cardie. 2003. Weakly supervised natural language learning without redundant views. In *HLT-NAACL*.

K. Nigam and R. Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *CIKM*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *EMNLP-CoNLL*.

M. A. Paskin. 2001. Grammatical bigrams. In *NIPS*.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL*.

R Development Core Team, 2011. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *ACL-IJCNLP*.

K. Rose. 1998. Deterministic annealing for clustering, compression, classification, regression and related optmization problems. *Proceedings of the IEEE*, 86.

A. Sarkar. 2001. Applying co-training methods to statistical parsing. In *NAACL*.

R. Schwartz, O. Abend, R. Reichart, and A. Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *ACL*.

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *ACL*.

N. A. Smith and J. Eisner. 2005a. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*.

N. A. Smith and J. Eisner. 2005b. Guiding unsupervised grammar induction using contrastive estimation. In *IJCAI: Grammatical Inference Applications*.

A. Søgaard and C. Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *COLING*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2009. Baby Steps: How "Less is More" in unsupervised dependency parsing. In *NIPS: Grammar Induction, Representation of Language and Language Learning*.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010a. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

V. I. Spitkovsky, D. Jurafsky, and H. Alshawi. 2010b. Profiting from mark-up: Hyper-text annotations for guided parsing. In *ACL*.

M. Stone. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B*, 36.

A. Yessenalina, Y. Yue, and C. Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *EMNLP*.

Y. Zhou and S. Goldman. 2004. Democratic co-learning. In *ICTAI*.

Unsupervised Dependency Parsing without Gold Part-of-Speech Tags

Valentin I. Spitkovsky[†*]
valentin@cs.stanford.edu

Hiyan Alshawi[*]
hiyan@google.com

Angel X. Chang[†*]
angelx@cs.stanford.edu

Daniel Jurafsky[‡†]
jurafsky@stanford.edu

[†]Computer Science Department
Stanford University
Stanford, CA, 94305

[*]Google Research
Google Inc.
Mountain View, CA, 94043

[‡]Department of Linguistics
Stanford University
Stanford, CA, 94305

Abstract

We show that categories induced by unsupervised word clustering can surpass the performance of gold part-of-speech tags in dependency grammar induction. Unlike classic clustering algorithms, our method allows a word to have different tags in different contexts. In an ablative analysis, we first demonstrate that this context-dependence is crucial to the superior performance of gold tags — requiring a word to always have the same part-of-speech significantly degrades the performance of manual tags in grammar induction, eliminating the advantage that human annotation has over unsupervised tags. We then introduce a sequence modeling technique that combines the output of a word clustering algorithm with context-colored noise, to allow words to be tagged differently in different contexts. With these new induced tags as input, our state-of-the-art dependency grammar inducer achieves 59.1% directed accuracy on Section 23 (all sentences) of the Wall Street Journal (WSJ) corpus — 0.7% higher than using gold tags.

1 Introduction

Unsupervised learning — machine learning without manually-labeled training examples — is an active area of scientific research. In natural language processing, unsupervised techniques have been successfully applied to tasks such as word alignment for machine translation. And since the advent of the web, algorithms that induce structure from unlabeled data have continued to steadily gain importance. In this paper we focus on unsupervised part-of-speech tagging and dependency parsing — two related problems of syntax discovery. Our methods are applicable to vast quantities of unlabeled monolingual text.

Not all research on these problems has been fully unsupervised. For example, to the best of our knowledge, every new state-of-the-art dependency grammar inducer since Klein and Manning (2004) relied on gold part-of-speech tags. For some time, multi-point performance degradations caused by switching to automatically induced word categories have been interpreted as indications that "good enough" parts-of-speech induction methods exist, justifying the focus on grammar induction with supervised part-of-speech tags (Bod, 2006), pace (Cramer, 2007). One of several drawbacks of this practice is that it weakens any conclusions that could be drawn about how computers (and possibly humans) learn in the absence of explicit feedback (McDonald et al., 2011).

In turn, not all unsupervised taggers actually induce word categories: Many systems — known as part-of-speech *disambiguators* (Merialdo, 1994) — rely on external dictionaries of possible tags. Our work builds on two older part-of-speech *inducers* — word clustering algorithms of Clark (2000) and Brown et al. (1992) — that were recently shown to be more robust than other well-known fully unsupervised techniques (Christodoulopoulos et al., 2010).

We investigate which properties of gold part-of-speech tags are useful in grammar induction and parsing, and how these properties could be introduced into induced tags. We also explore the number of word classes that is good for grammar induction: in particular, whether categorization is needed at all. By removing the "unrealistic simplification" of using gold tags (Petrov et al., 2011, §3.2, Footnote 4), we will go on to demonstrate why grammar induction from plain text is no longer "still too difficult."

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1281–1290,
Edinburgh, Scotland, UK, July 27–31, 2011.

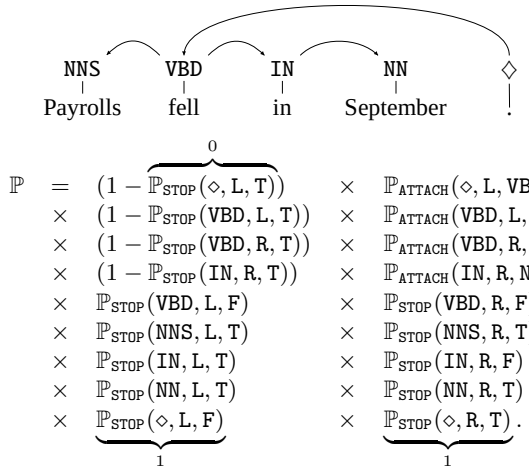


Figure 1: A dependency structure for a short WSJ sentence and its probability, factored by the DMV, using gold tags, after summing out $\mathbb{P}_{\text{ORDER}}$ (Spitkovsky et al., 2009).

2 Methodology

In all experiments, we model the English grammar via Klein and Manning's (2004) Dependency Model with Valence (DMV), induced from subsets of not-too-long sentences of the Wall Street Journal (WSJ).

2.1 The Model

The original DMV is a single-state head automata model (Alshawi, 1996) over lexical word classes $\{c_w\}$ — gold part-of-speech tags. Its generative story for a sub-tree rooted at a head (of class c_h) rests on three types of independent decisions: (i) initial direction $dir \in \{\text{L}, \text{R}\}$ in which to attach children, via probability $\mathbb{P}_{\text{ORDER}}(c_h)$; (ii) whether to seal dir, stopping with probability $\mathbb{P}_{\text{STOP}}(c_h, dir, adj)$, conditioned on $adj \in \{\text{T}, \text{F}\}$ (true iff considering dir's first, i.e., *adjacent*, child); and (iii) attachments (of class c_a), according to $\mathbb{P}_{\text{ATTACH}}(c_h, dir, c_a)$. This recursive process produces only projective trees. A root token $\diamond$ generates the head of the sentence as its left (and only) child (see Figure 1 for a simple, concrete example).

2.2 Learning Algorithms

The DMV lends itself to unsupervised learning via inside-outside re-estimation (Baker, 1979). Klein and Manning (2004) initialized their system using an "ad-hoc harmonic" completion, followed by training using 40 steps of EM (Klein, 2005). We reproduce this set-up, iterating without actually verifying convergence, in most of our experiments (#1–4, §3–4).

Experiments #5–6 (§5) employ our new state-of-the-art grammar inducer (Spitkovsky et al., 2011), which uses constrained Viterbi EM (details in §5).

2.3 Training Data

The DMV is usually trained on a customized subset of Penn English Treebank's Wall Street Journal portion (Marcus et al., 1993). Following Klein and Manning (2004), we begin with reference constituent parses, prune out all empty sub-trees and remove punctuation and terminals (tagged # and $) that are not pronounced where they appear. We then train only on the remaining sentence *yields* consisting of no more than fifteen tokens (WSJ15), in most of our experiments (#1–4, §3–4); by contrast, Klein and Manning's (2004) original system was trained using less data: sentences up to length ten (WSJ10).[1]

Our final experiments (#5–6, §5) employ a simple scaffolding strategy (Spitkovsky et al., 2010a) that follows up initial training at WSJ15 ("less is more") with an additional training run ("leapfrog") that incorporates most sentences of the data set, at WSJ45.

2.4 Evaluation Methods

Evaluation is against the training set, as is standard practice in unsupervised learning, in part because Klein and Manning (2004, §3) did not smooth the DMV (Klein, 2005, §6.2). For most of our experiments (#1–4, §3–4), this entails starting with the reference trees from WSJ15 (as modified in §2.3), automatically converting their labeled constituents into unlabeled dependencies using deterministic "head-percolation" rules (Collins, 1999), and then computing (directed) dependency accuracy scores of the corresponding induced trees. We report overall percentages of correctly guessed arcs, including the arcs from sentence root symbols, as is standard practice (Paskin, 2001; Klein and Manning, 2004).

For a meaningful comparison with previous work, we also test some of the models from our earlier experiments (#1,3) — and both models from final experiments (#5,6) — against Section 23 of WSJ$^\infty$, after applying Laplace (a.k.a. "add one") smoothing.

[1] WSJ15 contains 15,922 sentences up to length fifteen (a total of 163,715 tokens, not counting punctuation) — versus 7,422 sentences of at most ten words (only 52,248 tokens) comprising WSJ10 — and is a better trade-off between the quantity and complexity of training data in WSJ (Spitkovsky et al., 2009).

	Accuracy		Viable
1. **manual** tags	*Unsupervised*	*Sky*	*Groups*
gold	50.7	*78.0*	36
mfc	**47.2**	74.5	34
mfp	40.4	76.4	160
ua	44.3	**78.4**	328
2. tagless **lexicalized** models			
full	25.8	**97.3**	49,180
partial	29.3	60.5	176
none	**30.7**	*24.5*	1
3. tags from a **flat** (Clark, 2000) clustering			
	47.8	**83.8**	197
4. prefixes of a **hierarchical** (Brown et al., 1992) clustering			
first 7 bits	46.4	73.9	96
8 bits	**48.0**	77.8	165
9 bits	46.8	**82.3**	262

Table 1: Directed accuracies for the "less is more" DMV, trained on WSJ15 (after 40 steps of EM) and evaluated also against WSJ15, using various lexical categories in place of gold part-of-speech tags. For each tag-set, we include its effective number of (non-empty) categories in WSJ15 and the oracle skylines (supervised performance).

3 Motivation and Ablative Analyses

The concepts of polysemy and synonymy are of fundamental importance in linguistics. For words that can take on multiple parts of speech, knowing the gold tag can reduce ambiguity, improving parsing by limiting the search space. Furthermore, pooling the statistics of words that play similar syntactic roles, as signaled by shared gold part-of-speech tags, can simplify the learning task, improving generalization by reducing sparsity. We begin with two sets of experiments that explore the impact that each of these factors has on grammar induction with the DMV.

3.1 Experiment #1: Human-Annotated Tags

Our first set of experiments attempts to isolate the effect that replacing gold part-of-speech tags with deterministic *one class per word* mappings has on performance, quantifying the cost of switching to a monosemous clustering (see Table 1: manual; and Table 4). Grammar induction with gold tags scores 50.7%, while the oracle skyline (an ideal, supervised instance of the DMV) could attain 78.0% accuracy.

It may be worth noting that only 6,620 (13.5%) of 49,180 unique tokens in WSJ appear with multiple part-of-speech tags. Most words, like *it*, are always tagged the same way (5,768 times PRP). Some words, like *gains*, usually serve as one part of speech (227 times NNS, as in *the gains*) but are occasionally used differently (5 times VBZ, as in *he gains*). Only 1,322 tokens (2.7%) appear with three or more different gold tags. However, this minority includes the most frequent word — *the* (50,959 times DT, 7 times JJ, 6 times NNP and once as each of CD, NN and VBP).[2]

token	*mfc*	*mfp*	*ua*
it	{PRP}	{PRP}	{PRP}
gains	{NNS}	{VBZ, NNS}	{VBZ, NNS}
the	{DT}	{JJ, DT}	{VBP, NNP, NN, JJ, DT, CD}

Table 2: Example most frequent class, most frequent pair and union all reassignments for tokens *it*, *the* and *gains*.

We experimented with three natural reassignments of part-of-speech categories (see Table 2). The first, *most frequent class* (mfc), simply maps each token to its most common gold tag in the entire WSJ (with ties resolved lexicographically). This approach discards two gold tags (types PDT and RBR are not most common for any of the tokens in WSJ15) and costs about three-and-a-half points of accuracy, in both supervised and unsupervised regimes.

Another reassignment, *union all* (ua), maps each token to the set of all of its observed gold tags, again in the entire WSJ. This inflates the number of groupings by nearly a factor of ten (effectively lexicalizing the most ambiguous words),[3] yet improves the oracle skyline by half-a-point over actual gold tags; however, learning is harder with this tag-set, losing more than six points in unsupervised training.

Our last reassignment, *most frequent pair* (mfp), allows up to two of the most common tags into a token's label set (with ties, once again, resolved lexicographically). This intermediate approach performs strictly worse than *union all*, in both regimes.

3.2 Experiment #2: Lexicalization Baselines

Our next set of experiments assesses the benefits of categorization, turning to lexicalized baselines that avoid grouping words altogether. All three models discussed below estimated the DMV *without* using the gold tags in any way (see Table 1: lexicalized).

[2] Some of these are annotation errors in the treebank (Banko and Moore, 2004, Figure 2): such (mis)taggings can severely degrade the accuracy of part-of-speech disambiguators, without additional supervision (Banko and Moore, 2004, §5, Table 1).

[3] Kupiec (1992) found that the 50,000-word vocabulary of the Brown corpus similarly reduces to ∼400 ambiguity classes.

First, not surprisingly, a fully-lexicalized model over nearly 50,000 unique words is able to essentially memorize the training set, supervised. (Without smoothing, it is possible to deterministically attach most rare words in a dependency tree correctly, etc.) Of course, local search is unlikely to find good instantiations for so many parameters, causing unsupervised accuracy for this model to drop in half.

For our next experiment, we tried an intermediate, partially-lexicalized approach. We mapped frequent words — those seen at least 100 times in the training corpus (Headden et al., 2009) — to their own individual categories, lumping the rest into a single "unknown" cluster, for a total of under 200 groups. This model is significantly worse for supervised learning, compared even with the monosemous clusters derived from gold tags; yet it is only slightly more learnable than the broken fully-lexicalized variant.

Finally, for completeness, we trained a model that maps every token to the same one "unknown" category. As expected, such a trivial "clustering" is ineffective in supervised training; however, it outperforms both lexicalized variants unsupervised,[4] strongly suggesting that lexicalization alone may be insufficient for the DMV and hinting that some degree of categorization is essential to its learnability.

Cluster #173		*Cluster #188*	
1.	open	1.	get
2.	free	2.	make
3.	further	3.	take
4.	higher	4.	find
5.	lower	5.	give
6.	similar	6.	keep
7.	leading	7.	pay
8.	present	8.	buy
9.	growing	9.	win
10.	increased	10.	sell
⋮		⋮	
37.	cool	42.	improve
⋮		⋮	
1,688.	up-wind	2,105.	zero-out

Table 3: Representative members for two of the flat word groupings: cluster #173 (left) contains adjectives, especially ones that take comparative (or other) complements; cluster #188 comprises bare-stem verbs (infinitive stems). (Of course, many of the words have other syntactic uses.)

[4]Note that it also beats supervised training. That isn't a bug: Spitkovsky et al. (2010b, §7.2) explain this paradox in the DMV.

4 Grammars over Induced Word Clusters

We have demonstrated the need for grouping similar words, estimated a bound on performance losses due to monosemous clusterings and are now ready to experiment with induced part-of-speech tags. We use two sets of established, publicly-available hard clustering assignments, each computed from a much larger data set than WSJ (approximately a million words). The first is a flat mapping (200 clusters) constructed by training Clark's (2000) distributional similarity model over several hundred million words from the British National and the English Gigaword corpora.[5] The second is a hierarchical clustering — binary strings up to eighteen bits long — constructed by running Brown et al.'s (1992) algorithm over 43 million words from the BLLIP corpus, minus WSJ.[6]

4.1 Experiment #3: A Flat Word Clustering

Our main purely unsupervised results are with a flat clustering (Clark, 2000) that groups words having similar context distributions, according to Kullback-Leibler divergence. (A word's context is an ordered pair: its left- and right-adjacent neighboring words.)

To avoid overfitting, we employed an implementation from previous literature (Finkel and Manning, 2009). The number of clusters (200) and the sufficient amount of training data (several hundred-million words) were tuned to a task (NER) that is not directly related to dependency parsing. (Table 3 shows representative entries for two of the clusters.)

We added one more category (#0) for unknown words. Now every token in WSJ could again be replaced by a coarse identifier (one of at most 201, instead of just 36), in both supervised and unsupervised training. (Our training code did not change.)

The resulting supervised model, though not as good as the fully-lexicalized DMV, was more than five points more accurate than with gold part-of-speech tags (see Table 1: flat). Unsupervised accuracy was lower than with gold tags (see also Table 4) but higher than with *all* three derived hard assignments. This suggests that polysemy (i.e., ability to

[5]`http://nlp.stanford.edu/software/stanford-postagger-2008-09-28.tar.gz:models/egw.bnc.200`

[6]`http://people.csail.mit.edu/maestro/papers/bllip-clusters.gz`

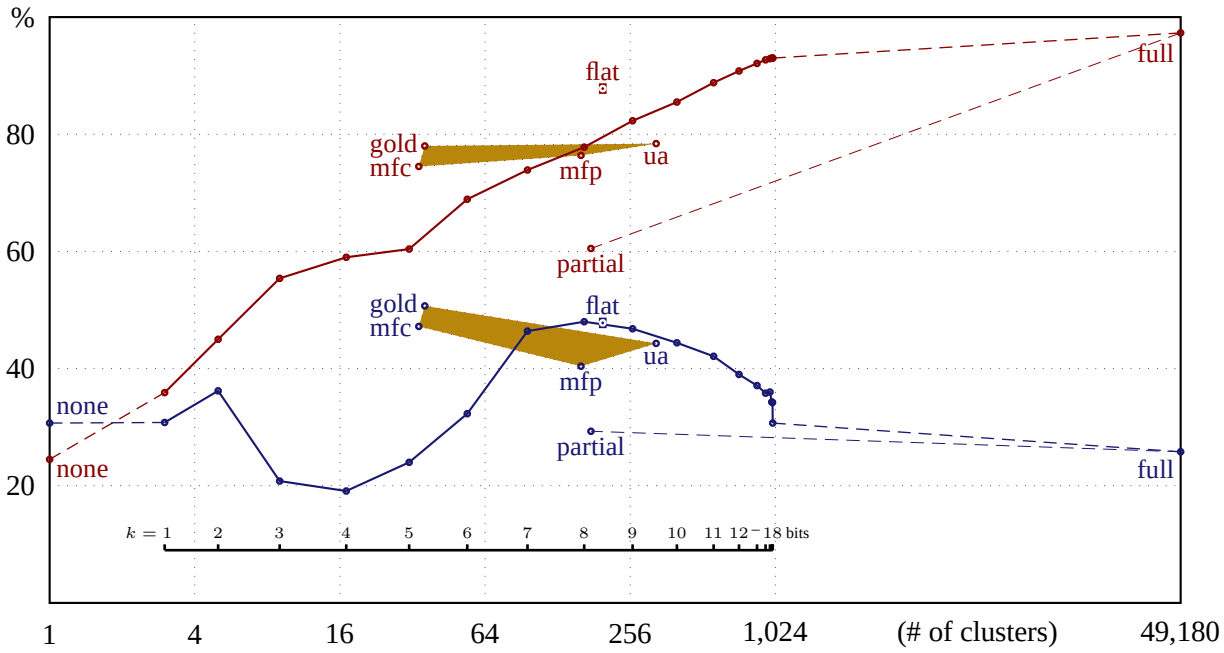


Figure 2: Parsing performance (accuracy on WSJ15) as a "function" of the number of syntactic categories, for all prefix lengths — $k \in \{1, \ldots, 18\}$ — of a hierarchical (Brown et al., 1992) clustering, connected by solid lines (dependency grammar induction in blue; supervised oracle skylines in red, above). Tagless lexicalized models (*full*, *partial* and *none*) connected by dashed lines. Models based on *gold* part-of-speech tags, and derived monosemous clusters (*mfc*, *mfp* and *ua*), shown as vertices of gold polygons. Models based on a *flat* (Clark, 2000) clustering indicated by squares.

tag a word differently in context) may be the primary advantage of manually constructed categorizations.

4.2 Experiment #4: A Hierarchical Clustering

The purpose of this batch of experiments is to show that Clark's (2000) algorithm isn't unique in its suitability for grammar induction. We found that Brown et al.'s (1992) older information-theoretic approach, which does not explicitly address the problems of rare and ambiguous words (Clark, 2000) and was designed to induce large numbers of plausible syntactic *and* semantic clusters, can perform just as well.

Once again, the sufficient amount of data (43 million words) was tuned in earlier work (Koo, 2010). His task of interest was, in fact, dependency parsing. But since this algorithm is hierarchical (i.e., there isn't a parameter for the number of categories), we doubt that there was a strong enough risk of overfitting to question the clustering's unsupervised nature.

As there isn't a set number of categories, we used binary prefixes of length k from each word's address in the computed hierarchy as cluster labels. Results for $7 \leq k \leq 9$ bits (approximately 100–250 non-empty clusters, close to the 200 we used before) are similar to those of flat clusters (see Table 1: hierarchical). Outside of this range, however, performance can be substantially worse (see Figure 2), consistent with earlier findings: Headden et al. (2008) demonstrated that (constituent) grammar induction, using the singular-value decomposition (SVD-based) tagger of Schütze (1995), also works best with 100–200 clusters. Important future research directions may include learning to automatically select a good number of word categories (in the case of flat clusterings) and ways of using multiple clustering assignments, perhaps of different granularities/resolutions, in tandem (e.g., in the case of a hierarchical clustering).

4.3 Further Evaluation

It is important to enable easy comparison with previous and future work. Since WSJ15 is not a standard test set, we evaluated two key experiments — "less is more" with gold part-of-speech tags (#1, Table 1: gold) and with Clark's (2000) clusters (#3, Table 1: flat) — on all sentences (not just length fifteen and shorter), in Section 23 of WSJ (see Table 4). This required smoothing both final models (§2.4).

We showed that two classic unsupervised word

	System Description		*Accuracy*
#1 (§3.1)	"less is more"	(Spitkovsky et al., 2009)	44.0
#3 (§4.1)	"less is more" with monosemous induced tags		41.4 (-2.6)

Table 4: Directed accuracies on Section 23 of WSJ (all sentences) for two experiments with the base system.

clusterings — one flat and one hierarchical — can be better for dependency grammar induction than monosemous syntactic categories derived from gold part-of-speech tags. And we confirmed that the unsupervised tags are worse than the actual gold tags, in a simple dependency grammar induction system.

5 State-of-the-Art without Gold Tags

Until now, we have deliberately kept our experimental methods simple and nearly identical to Klein and Manning's (2004), for clarity. Next, we will explore how our main findings generalize beyond this toy setting. A preliminary test will simply quantify the effect of replacing gold part-of-speech tags with the monosemous flat clustering (as in experiment #3, §4.1) on a modern grammar inducer. And our last experiment will gauge the impact of using a polysemous (but still unsupervised) clustering instead, obtained by executing standard sequence labeling techniques to introduce context-sensitivity into the original (independent) assignment or words to categories.

These final experiments are with our latest state-of-the-art system (Spitkovsky et al., 2011) — a partially lexicalized extension of the DMV that uses constrained Viterbi EM to train on nearly all of the data available in WSJ, at WSJ45 (48,418 sentences; 986,830 non-punctuation tokens). The key contribution that differentiates this model from its predecessors is that it incorporates punctuation into grammar induction (by turning it into parsing constraints, instead of ignoring punctuation marks altogether). In training, the model makes a simplifying assumption — that sentences can be split at punctuation and that the resulting fragments of text could be parsed independently of one another (these parsed fragments are then reassembled into full sentence trees, by parsing the sequence of their own head words). Furthermore, the model continues to take punctuation marks into account in inference (using weaker, more accurate constraints, than in training). This system scores 58.4% on Section 23 of WSJ$^\infty$ (see Table 5).

5.1 Experiment #5: A Monosemous Clustering

As in experiment #3 (§4.1), we modified the base system in exactly one way: we swapped out gold part-of-speech tags and replaced them with a flat distributional similarity clustering. In contrast to simpler models, which suffer multi-point drops in accuracy from switching to unsupervised tags (e.g., 2.6%), our new system's performance degrades only slightly, by 0.2% (see Tables 4 and 5). This result improves over substantial performance degradations previously observed for unsupervised dependency parsing with induced word categories (Klein and Manning, 2004; Headden et al., 2008, *inter alia*).[7]

One risk that arises from using gold tags is that newer systems could be finding cleverer ways to exploit manual labels (i.e., developing an over-reliance on gold tags) instead of actually learning to acquire language. Part-of-speech tags are *known* to contain significant amounts of information for unlabeled dependency parsing (McDonald et al., 2011, §3.1), so we find it reassuring that our latest grammar inducer is *less* dependent on gold tags than its predecessors.

5.2 Experiment #6: A Polysemous Clustering

Results of experiments #1 and 3 (§3.1, 4.1) suggest that grammar induction stands to gain from relaxing the *one class per word* assumption. We next test this conjecture by inducing a polysemous unsupervised word clustering, then using it to induce a grammar.

Previous work (Headden et al., 2008, §4) found that simple bitag hidden Markov models, classically trained using the Baum-Welch (Baum, 1972) variant of EM (HMM-EM), perform quite well,[8] on average, across different grammar induction tasks. Such sequence models incorporate a sensitivity to context via state transition probabilities $\mathbb{P}_{\texttt{TRAN}}(t_i \mid t_{i-1})$, capturing the likelihood that a tag t_i immediately follows the tag t_{i-1}; emission probabilities $\mathbb{P}_{\texttt{EMIT}}(w_i \mid t_i)$ capture the likelihood that a word of type t_i is w_i.

[7] We also briefly comment on this result in the "punctuation" paper (Spitkovsky et al., 2011, §7), published concurrently.

[8] They are also competitive with Bayesian estimators, on larger data sets, with cross-validation (Gao and Johnson, 2008).

	System Description		*Accuracy*
(§5)	"punctuation"	(Spitkovsky et al., 2011)	58.4
#5 (§5.1)	"punctuation" with monosemous induced tags		58.2 (-0.2)
#6 (§5.2)	"punctuation" with **context-sensitive** induced tags		**59.1** (+**0.7**)

Table 5: Directed accuracies on Section 23 of WSJ (all sentences) for experiments with the state-of-the-art system.

We need a context-sensitive tagger, and HMM models are good — relative to other tag-inducers. However, they are not better than gold tags, at least when trained using a modest amount of data.[9] For this reason, we decided to relax the monosemous flat clustering, plugging it in as an initializer for the HMM. The main problem with this approach is that, at least without smoothing, every monosemous labeling is trivially at a local optimum, since $\mathbb{P}(t_i \mid w_i)$ is deterministic. To escape the initial assignment, we used a "noise injection" technique (Selman et al., 1994), inspired by the contexts of Clark (2000). First, we collected the MLE statistics for $\mathbb{P}_{\text{R}}(t_{i+1} \mid t_i)$ and $\mathbb{P}_{\text{L}}(t_i \mid t_{i+1})$ in WSJ, using the flat monosemous tags. Next, we replicated the text of WSJ 100-fold. Finally, we retagged this larger data set, as follows: with probability 80%, a word kept its monosemous tag; with probability 10%, we sampled a new tag from the left context ($\mathbb{P}_{\text{L}}$) associated with the original (monosemous) tag of its rightmost neighbor; and with probability 10%, we drew a tag from the right context ($\mathbb{P}_{\text{R}}$) of its leftmost neighbor.[10] Given that our initializer — and later the input to the grammar inducer — are hard assignments of tags to words, we opted for (the faster and simpler) Viterbi training.

In the spirit of reproducibility, we again used an off-the-shelf component for tagging-related work.[11] Viterbi training converged after just 17 steps, replacing the original monosemous tags for 22,280 (of 1,028,348 non-punctuation) tokens in WSJ. For example, the first changed sentence is #3 (of 49,208):

> *Some "circuit breakers" installed after the October 1987 crash failed their first test, traders say, unable to **cool** the selling panic in both stocks and futures.*

Above, the word *cool* gets relabeled as #188 (from #173 — see Table 3), since its context is more suggestive of an infinitive verb than of its usual grouping with adjectives. (A proper analysis of all changes, however, is beyond the scope of this work.)

Using this new context-sensitive hard assignment of tokens to unsupervised categories our grammar inducer attained a directed accuracy of 59.1%, nearly a full point better than with the monosemous hard assignment (see Table 5). To the best of our knowledge it is also the first state-of-the-art unsupervised dependency parser to perform better with induced categories than with gold part-of-speech tags.

6 Related Work

Early work in dependency grammar induction already relied on gold part-of-speech tags (Carroll and Charniak, 1992). Some later models (Yuret, 1998; Paskin, 2001, *inter alia*) attempted full lexicalization. However, Klein and Manning (2004) demonstrated that effort to be worse at recovering dependency arcs than choosing parse structures at random, leading them to incorporate gold tags into the DMV.

Klein and Manning (2004, §5, Figure 6) had also tested their own models with induced word classes, constructed using a distributional similarity clustering method (Schütze, 1995). Without gold part-of-speech tags, their combined DMV+CCM model was about five points worse, both in (directed) unlabeled dependency accuracy (42.3% vs. 47.5%)[12] and unlabeled bracketing F_1 (72.9% vs. 77.6%), on WSJ10.

In constituent parsing, earlier Seginer (2007a, §6, Table 1) built a fully-lexicalized grammar inducer

[9] All of Headden et al.'s (2008) grammar induction experiments with induced parts-of-speech were worse than their best results using gold part-of-speech tags, most likely because they used a very small corpus (half of WSJ10) to cluster words.

[10] We chose the sampling split (80:10:10) and replication parameter (100) somewhat arbitrarily, so better results could likely be obtained with tuning. However, we suspect that the real gains would come from using soft clustering techniques (Hinton and Roweis, 2003; Pereira et al., 1993, *inter alia*) and propagating (joint) estimates of tag distributions into a parser. Our ad-hoc approach is intended to serve solely as a proof of concept.

[11] David Elworthy's `C+ tagger`, with options `-i t -G -l`, available from `http://friendly-moose.appspot.com/code/NewCpTag.zip`.

[12] On the same evaluation set (WSJ10), our context-sensitive system without gold tags (Experiment #6, §5.2) scores 66.8%.

that was competitive with DMV+CCM despite not using gold tags. His CCL parser has since been improved via a "zoomed learning" technique (Reichart and Rappoport, 2010). Moreover, Abend et al. (2010) reused CCL's internal distributional representation of words in a cognitively-motivated part-of-speech inducer. Unfortunately their tagger did not make it into Christodoulopoulos et al.'s (2010) excellent and otherwise comprehensive evaluation.

Outside monolingual grammar induction, fully-lexicalized statistical dependency transduction models have been trained from unannotated parallel bitexts for machine translation (Alshawi et al., 2000). More recently, McDonald et al. (2011) demonstrated an impressive alternative to grammar induction by projecting reference parse trees from languages that have annotations to ones that are resource-poor.[13] It uses graph-based label propagation over a bilingual similarity graph for a sentence-aligned parallel corpus (Das and Petrov, 2011), inducing part-of-speech tags from a universal tag-set (Petrov et al., 2011).

Even in supervised parsing we are starting to see a shift away from using gold tags. For example, Alshawi et al. (2011) demonstrated good results for mapping text to underspecified semantics via dependencies without resorting to gold tags. And Petrov et al. (2010, §4.4, Table 4) observed only a small performance loss "going POS-less" in question parsing.

We are not aware of any systems that induce both syntactic trees and their part-of-speech categories. However, aside from the many systems that induce trees from gold tags, there are also unsupervised methods for inducing syntactic categories from gold trees (Finkel et al., 2007; Pereira et al., 1993), as well as for inducing dependencies from gold constituent annotations (Sangati and Zuidema, 2009; Chiang and Bikel, 2002). Considering that Headden et al.'s (2008) study of part-of-speech taggers found no correlation between standard tagging metrics and the quality of induced grammars, it may be time for a unified treatment of these very related syntax tasks.

[13] When the target language is English, however, their best accuracy (projected from Greek) is low: 45.7% (McDonald et al., 2011, §4, Table 2); tested on the same CoNLL 2007 evaluation set (Nivre et al., 2007), our "punctuation" system with context-sensitive induced tags (trained on WSJ45, without gold tags) performs substantially better, scoring 51.6%. Note that this is also an improvement over our system trained on the CoNLL set using gold tags: 50.3% (Spitkovsky et al., 2011, §8, Table 6).

7 Discussion and Conclusions

Unsupervised word clustering techniques of Brown et al. (1992) and Clark (2000) are well-suited to dependency parsing with the DMV. Both methods outperform gold parts-of-speech in supervised modes. And both can do better than monosemous clusters derived from gold tags in unsupervised training. We showed how Clark's (2000) flat tags can be relaxed, using context, with the resulting polysemous clustering outperforming gold part-of-speech tags for the English dependency grammar induction task.

Monolingual evaluation is a significant flaw in our methodology, however. One (of many) take-home points made in Christodoulopoulos et al.'s (2010) study is that results on one language do not necessarily correlate with other languages.[14] Assuming that our results do generalize, it will still remain to remove the present reliance on gold tokenization and sentence boundary labels. Nevertheless, we feel that eliminating gold tags is an important step towards the goal of fully-unsupervised dependency parsing.

We have cast the utility of a categorization scheme as a combination of two effects on parsing accuracy: a synonymy effect and a polysemy effect. Results of our experiments with both full and partial lexicalization suggest that grouping similar words (i.e., synonymy) is vital to grammar induction with the DMV. This is consistent with an established viewpoint, that simple tabulation of frequencies of words participating in certain configurations cannot be reliably used for comparing their likelihoods (Pereira et al., 1993, §4.2): "The statistics of natural languages is inherently ill defined. Because of Zipf's law, there is never enough data for a reasonable estimation of joint object distributions." Seginer's (2007b, §1.4.4) argument, however, is that the Zipfian distribution — a property of words, not parts-of-speech — should allow frequent words to successfully guide

[14] Furthermore, it would be interesting to know how sensitive different head-percolation schemes (Yamada and Matsumoto, 2003; Johansson and Nugues, 2007) would be to gold versus unsupervised tags, since the Magerman-Collins rules (Magerman, 1995; Collins, 1999) agree with gold dependency annotations only 85% of the time, even for WSJ (Sangati and Zuidema, 2009). Proper intrinsic evaluation of dependency grammar inducers is not yet a solved problem (Schwartz et al., 2011).

parsing and learning: "A relatively small number of frequent words appears almost everywhere and most words are never too far from such a frequent word (this is also the principle behind successful part-of-speech induction)." We believe that it is important to thoroughly understand how to reconcile these only seemingly conflicting insights, balancing them both in theory and in practice. A useful starting point may be to incorporate frequency information in the parsing models directly — in particular, capturing the relationships between words of various frequencies.

The polysemy effect appears smaller but is less controversial: Our experiments suggest that the primary drawback of the classic clustering schemes stems from their *one class per word* nature — and not a lack of supervision, as may be widely believed. Monosemous groupings, even if they are themselves derived from human-annotated syntactic categories, simply cannot disambiguate words the way gold tags can. By relaxing Clark's (2000) flat clustering, using contextual cues, we improved dependency grammar induction: directed accuracy on Section 23 (all sentences) of the WSJ benchmark increased from 58.2% to 59.1% — from slightly worse to better than with gold tags (58.4%, previous state-of-the-art).

Since Clark's (2000) word clustering algorithm is already context-sensitive in training, we suspect that one could do better simply by preserving the polysemous nature of its internal representation. Importing the relevant distributions into a sequence tagger directly would make more sense than going through an intermediate monosemous summary. And exploring other uses of *soft* clustering algorithms — perhaps as inputs to part-of-speech disambiguators — may be another fruitful research direction. We believe that a *joint* treatment of grammar and parts-of-speech induction could fuel major advances in both tasks.

Acknowledgments

Partially funded by the Air Force Research Laboratory (AFRL), under prime contract no. FA8750-09-C-0181, and by NSF, via award #IIS-0811974. We thank Omri Abend, Spence Green, David McClosky and the anonymous reviewers for many helpful comments on draft versions of this paper.

References

O. Abend, R. Reichart, and A. Rappoport. 2010. Improved unsupervised POS induction through prototype discovery. In *ACL*.

H. Alshawi, S. Bangalore, and S. Douglas. 2000. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26.

H. Alshawi, P.-C. Chang, and M. Ringgaard. 2011. Deterministic statistical mapping of sentences to underspecied semantics. In *IWCS*.

H. Alshawi. 1996. Head automata for speech translation. In *ICSLP*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*.

M. Banko and R. C. Moore. 2004. Part of speech tagging in context. In *COLING*.

L. E. Baum. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In *Inequalities*.

R. Bod. 2006. An all-subtrees approach to unsupervised parsing. In *COLING-ACL*.

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18.

G. Carroll and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University.

D. Chiang and D. M. Bikel. 2002. Recovering latent information in treebanks. In *COLING*.

C. Christodoulopoulos, S. Goldwater, and M. Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *EMNLP*.

A. Clark. 2000. Inducing syntactic categories by context distribution clustering. In *CoNLL-LLL*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

B. Cramer. 2007. Limitations of current grammar induction algorithms. In *ACL: Student Research*.

D. Das and S. Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *ACL*.

J. R. Finkel and C. D. Manning. 2009. Joint parsing and named entity recognition. In *NAACL-HLT*.

J. R. Finkel, T. Grenager, and C. D. Manning. 2007. The infinite tree. In *ACL*.

J. Gao and M. Johnson. 2008. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *EMNLP*.

W. P. Headden, III, D. McClosky, and E. Charniak. 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. In *COLING*.

W. P. Headden, III, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *NAACL-HLT*.

G. Hinton and S. Roweis. 2003. Stochastic neighbor embedding. In *NIPS*.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *NODALIDA*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

D. Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

T. Koo. 2010. *Advances in Discriminative Dependency Parsing*. Ph.D. thesis, MIT.

J. Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6.

D. M. Magerman. 1995. Statistical decision-tree models for parsing. In *ACL*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

R. McDonald, S. Petrov, and K. Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *EMNLP*.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *EMNLP-CoNLL*.

M. A. Paskin. 2001. Grammatical bigrams. In *NIPS*.

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *ACL*.

S. Petrov, P.-C. Chang, M. Ringgaard, and H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *EMNLP*.

S. Petrov, D. Das, and R. McDonald. 2011. A universal part-of-speech tagset. In *ArXiv*.

R. Reichart and A. Rappoport. 2010. Improved fully unsupervised parsing with zoomed learning. In *EMNLP*.

F. Sangati and W. Zuidema. 2009. Unsupervised methods for head assignments. In *EACL*.

H. Schütze. 1995. Distributional part-of-speech tagging. In *EACL*.

R. Schwartz, O. Abend, R. Reichart, and A. Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *ACL*.

Y. Seginer. 2007a. Fast unsupervised incremental parsing. In *ACL*.

Y. Seginer. 2007b. *Learning Syntactic Structure*. Ph.D. thesis, University of Amsterdam.

B. Selman, H. A. Kautz, and B. Cohen. 1994. Noise strategies for improving local search. In *AAAI*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2009. Baby Steps: How "Less is More" in unsupervised dependency parsing. In *NIPS: Grammar Induction, Representation of Language and Language Learning*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010a. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *NAACL-HLT*.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011. Punctuation: Making a point in unsupervised dependency parsing. In *CoNLL*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *IWPT*.

D. Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, MIT.

Exploiting Syntactic and Distributional Information for Spelling Correction with Web-Scale N-gram Models

Wei Xu[c,*] **Joel Tetreault**[a] **Martin Chodorow**[b] **Ralph Grishman**[c] **Le Zhao**[d]
[a]Educational Testing Service, Princeton, NJ, USA
jtetreault@ets.org
[b]Hunter College of CUNY, New York, NY, USA
martin.chodorow@hunter.cuny.edu
[c]New York University, NY, USA
{xuwei,grishman}@cs.nyu.edu
[d]Carnegie Mellon University, Pittsburgh, PA, USA
lezhao@cs.cmu.edu

Abstract

We propose a novel way of incorporating dependency parse and word co-occurrence information into a state-of-the-art web-scale n-gram model for spelling correction. The syntactic and distributional information provides extra evidence in addition to that provided by a web-scale n-gram corpus and especially helps with data sparsity problems. Experimental results show that introducing syntactic features into n-gram based models significantly reduces errors by up to 12.4% over the current state-of-the-art. The word co-occurrence information shows potential but only improves overall accuracy slightly.

1 Introduction

The function of context-sensitive text correction is to identify word-choice errors in text (Bergsma et al., 2009). It can be viewed as a lexical disambiguation task (Lapata and Keller, 2005), where a system selects from a predefined confusion word set, such as {affect, effect} or {complement, compliment}, and provides the most appropriate word choice given the context. Typically, one determines if a word has been used correctly based on lexical, syntactic and semantic information from the context of the word. One of the top performing models of spelling correction (Bergsma et al., 2010) is based on web-scale n-gram counts, which reflect both syntax and meaning. However, even with a large-scale n-gram corpus, data sparsity can hurt performance in two ways. First, n-gram based methods require exact word and order matches. If there is a low frequency word in the context, such as a person's name, there will be little, if any, evidence in the n-gram data to support the usage. Second, if the target confusable word is rare, there will not be enough n-gram support or training data to render a confident decision. Because of the data sparsity problem, language modeling is not always sufficient to capture the meaning of the sentence and the correct usage of the word.

Take a sentence from The New York Times (NYT) for example: "'This fellow's won a war,' the dean of the capital's press corps, David Broder, announced on 'Meet the Press' after **complimenting** the president on the 'great sense of authority and command' he exhibited in a flight suit." Unfortunately, neither the phrase "complementing the president" nor "complimenting the president" exists in the web-scale Google N-gram corpus (Brants and Franz, 2006). The n-gram models decide solely based on the frequency of the bi-grams "after comple(i)menting" and "comple(i)menting the", which are common usages for both words. The real question is whether we are more likely to "compliment" or "complement" a person, the "president". Several clues could help us answer that question. A dependency parser can identify the word "president" as the subject of "compliment" or "complement" which also may be the case in some of the training data. Lexical co-occurrence (Edmonds, 1997) and semantic word relatedness measurements, such as Random Indexing (Sahlgren, 2006), could provide evidence that "compliment" is more likely to co-occur with "president" than "complement". Fur-

*This work was done when the first author was an intern for Educational Testing Service.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1291–1300,
Edinburgh, Scotland, UK, July 27–31, 2011.

thermore, some important clues can be quite distant from the target word, e.g. outside the 9-word context window Bergsma et al. (2010) and Carlson (2007) used. Consider another sentence in the NYT corpus, "GM says the addition of OnStar, which includes a system that automatically notifies an OnStar operator if the vehicle is involved in a collision, **complements** the Vue's top five-star safety rating for the driver and front passenger in both front- and side-impact crash tests." The dependency parser finds the object of "complement" is "rating", which is outside the 9-word window.

We propose enhancing state-of-the-art web-scale n-gram models for spelling correction with syntactic structures and distributional information. For our work, we build on a baseline system that combines n-gram and lexical features (Bergsma et al., 2010). Specifically, this paper makes the following contributions:

1. We show that the baseline system can be improved by augmenting it with dependency parse features.

2. We show that the impact of parse features can be further augmented when combined with distributional information, specifically word co-occurrence information.

In the following section, we describe related work and how our approach differs from these approaches. In Sections 3 and 4, we discuss our methods for using parse features and word co-occurrence information. In Section 5, we present experimental results and analysis.

2 Related Work

A variety of approaches have been proposed for context-sensitive spelling correction ranging from semantic methods to machine learning classifiers to large-scale n-gram models.

Some semantics-based systems have been developed based on an intuitive assumption that the intended word is more likely to be semantically coherent with the context than is a spelling error. Jones and Martin (1997) made use of the semantic similarity produced by Latent Semantic Analysis. Budanitsky and Hirst (2001) investigated the effectiveness of predicting words based on different semantic similarity/distance measures in WordNet. Both systems report performance that is lower than systems developed more recently.

A variety of machine-learning methods have been proposed in spelling correction and preposition and article error correction fields, such as Bayesian classifiers (Golding, 1995; Golding and Roth, 1996), Winnow-based learning (Golding and Roth, 1999), decision lists (Golding, 1995), transformation-based learning (Mangu and Brill, 1997), augmented mixture models (Cucerzan and Yarowsky, 2002) and maximum entropy classifiers (Izumi et al., 2003; Han et al., 2006; Chodorow et al., 2007; Tetreault and Chodorow, 2008; Felice and Pulman, 2008). Despite their differences, these approaches mainly use contextual features to capture the lexical, semantic and/or syntactic environment of the target word.

The use of distributional similarity measures for spelling correction has been previously explored in (Mohammad and Hist, 2006). In our work, distributional similarity is not the primary contribution but we show the impact it can have when used in conjunction with a large scale n-gram model and with parse features, which allows the system to select words outside the local window for distributional similarity. In the prior work, the words for distributional similarity are constrained to the local window, and positional information of the words is not encoded.

Recent work (Carlson and Fette, 2007; Gamon et al., 2008; Bergsma et al., 2009) has demonstrated that large-scale language modeling is extremely helpful for contextual spelling correction and other lexical disambiguation tasks. These systems make the word choice depending on how frequently each candidate word has been seen in the given context in web-scale data. As n-gram data has become more readily available, such as the Google N-gram Corpus, the likelihood of a word being used in a certain context can be better estimated.

Bergsma et al. (2009; 2010) presented a series of simple but powerful models which relied heavily on web-scale n-gram counts. From the Google Web N-gram Corpus, they retrieve counts of n-grams of different sizes (2-5) and positions that span the target word w0 within a window of 9 words. For example, for the following sentence: "The system tried to decide {among, between} the two confus-

able words.", the method would extract the five 5-gram patterns, shown below in Figure 2, where $w0$ can be either word in the confusion set {among, between} in this particular example. Similarly, there are four 4-grams, three 3-grams, and two 2-grams, in total, 14 n-grams for each of the words in the confusion set.

system tried to decide $w0$
tried to decide $w0$ the
to decide $w0$ the two
decide $w0$ the two confusable
$w0$ the two confusable words

We briefly describe three of Bergsma et al.'s (2009; 2010) best systems below, which are reported to achieve state-of-the-art accuracy (NG = n-gram; LEX = lexical).

1. **sumLM**: For each candidate word, (Bergsma et al., 2009) sum the log-counts of all 14 patterns filled with the candidate, and choose the candidate with the highest total.

2. **NG**: Bergsma et al. (2009) exploit each candidate's 14 log-counts of n-gram patterns as features in a Support Vector Machine (SVM) model.

3. **NG+LEX**: Bergsma et al. (2010) augment the NG model with lexical features (described in detail in Section 3.1).

Bergsma et al. (2009; 2010) restricted their experiments to only five confusion sets where the reported performance in (Golding and Roth, 1999) was below 90%: {among, between}, {amount, number}, {cite, sight, site}, {peace, piece} and {raise, rise}. They reported that the SVM model with NG features outperformed its unsupervised version, sumLM. However, the limited confusion word sets they evaluated may not comprehensively represent the word usage errors that writers typically make. In this paper, we test nine additional commonly confused word pairs to expand the scope of the evaluation. These words were selected based on their lower frequencies compared to the five pairs in the above work (as shown later in Table 2).

3 Enhanced N-gram Models with Parse Features

To our knowledge, only (Elmi and Evans, 1998) have used parsing for spell correction. They focus on using a parser as a filter to discriminate between possible real-world corrections where the part-of-speech differs. In our work, we show that parse features are effective when used directly in the classification mode (as opposed to as a final filter) to select the best correction regardless of whether or not the part-of-speech of the choices differ.

Statistical parsers have also seen limited use in the sister tasks of preposition and article error detection (Hermet et al., 2008; Lee and Knutsson, 2008; Felice and Pulman, 2009; Tetreault et al., 2010) and verb sense disambiguation (Dligach and Palmer, 2008). In those instances where parsers have been used, they have mainly provided shallow analyses or relations involving specific target words, such as a preposition or verb. Unlike preposition errors, spelling errors can occur in any word.

In this paper, we propose a novel way to incorporate the parse into spelling correction, applying the parser to sentences filled by each candidate word equivalently and extracting salient features. This overcomes two problem in the existing methods: 1) the parse trees of the same sentence filled by different confusion words can be different. However, in the test phase, we do not know which word should be put in the sentences to create parse features for test examples. Previous studies (Tetreault et al., 2010) failed to discuss this issue. 2) Some existing work (Whitelaw et al., 2009; Rozovskaya and Roth, 2010) in the text correction field introduced artificial errors into training data to adapt the system to better handle ill-formed text. But this method will encounter serious data sparsity problems when facing rare words.

3.1 Baseline System

We chose one of the leading spelling correction systems, (Bergsma et al., 2010), as our primary baseline. As noted earlier, it is an SVM-based system combining web-scale n-gram counts (NG) and contextual words (LEX) as features. To simplify the explanation, throughout the paper, we will only consider the situation with two confusion words. The

problem with more than two words in pre-defined confusion sets can be solved similarly by using a one-vs.-all strategy. As we mentioned in Section 2, NG features include log-counts of 3-to-5-gram patterns for each candidate word with the given context. LEX features can be broken down into three subcategories: 1) bag-of-words (words at all positions in a 9-word window around the target word), 2) indicators for the words preceding or following the target word, and 3) indicators for all n-grams and their positions. For the sentence "The system tried to decide {among, between} the two confusable words.", examples of bag-of-word features would be "tried", "two", etc., the two positional bigrams would be "decide" and "the", and examples of the n-gram features would be right-trigram = "among the two" and left-4-gram = "tried to decide between".

3.2 Parse Features

The benefit of introducing dependency parse features is that 1) parse features capture contextual information in a larger context window; 2) parse features specify which words in the context are salient to the usage of the target word while purely lexically based approaches treat all words in the context equally. We use the Stanford dependency parser (de Marneffe et al., 2006) to extract six relevant feature classes.

Parse Features (PAR):

1. relation names (target word as head)
2. complement of the target word
3. combination of 1 and 2
4. relation names (target word as complement)
5. head of the target word
6. combination of 4 and 5

Each of these six classes of PAR features can contain zero to many values, since the target word can be involved in none to multiple grammatical relations and features of different filler words are merged together. The PAR features, like the LEX features, are binary. In Table 1, we present the parse features for an example sentence. The parse features here are listed as string values, but are later converted into binary numbers in the vectors for the SVM model.

4 Distributional Word Co-occurrence

Though lexical and parse features are complementary to n-gram models, they are learned from a normal training corpus and may not have enough coverage due to data sparsity. Take a sentence from the NYT for example: "An economist, he began his career as a professor – he is still called 'the professor,' by friends as a compliment and by foes as an insult – and taught at Harvard and Stanford ." If the most indicative word "friends" does not appear or does not appear enough times in the local context or dependencies with "compliment" as compared to "complement" in the training corpus, then the classifier may be unable to make the correct selection.

It is impractical and computationally costly to enlarge the training corpus without limit to include all possible language phenomena. A good compromise is to use word co-occurrence information from web-scale data. The other option is to make use of high-order word co-occurrence, which is included in many semantic word relatedness measures, such as Latent Semantic Analysis (LSA) (Landauer et al., 1998; Deerwester et al., 1990) or Random Indexing, both of which can be estimated from a moderate-size corpus.

Our intuition is to choose the confusion word which is most relevant to a given context. We define the salient words in context as a set M=m1, m2, m3, ..., and the relevance between two words as a function Relevance(w1, w2), which can either be calculated from word co-occurrence or Random Indexing. The score of each candidate word c in the confusion set given a context with meaningful words M is calculated by the following formula:

$$Score(c) = \sum_{m \in M} Relevance(c, m)$$

In this paper, we experiment with first-order word co-occurrence and Random Indexing as relevance measures. And we define salient contextual words as heads or complements in the dependency relations with the target word. In this way, we use the parse information to constrain the two distribution models. Thus the word co-occurrence information

Feature Name	PAR Features (compliment)	PAR Features (complement)
1. Head Relation Name	ccomp	appos
2. Head of Relation	says	collisions
3. Head Combination	ccomp_says	appos_collisions
4. Comp Relation Name	nsubj	dep
5. Comp of Relation	addition	rating
6. Comp Combination	nsub_addition	dep_rating

Table 1: Parse Feature Example for the sentence: "GM says the addition of OnStar, which includes a system that automatically notifies an OnStar operator if the vehicle is involved in a collision, **complements** the Vue's top five-star safety rating for the driver and front passenger in both front- and side-impact crash tests."

considerably overlaps with some values of the PAR features, but provides extra evidence from web-scale data rather than a limited amount of training data.

4.1 First-order Word Co-occurrence

The relevance based on first-order word co-occurrence is calculated from the Google Web 5-gram Corpus in a fashion similar to how we dealt with n-gram counts in the previous section. Given two words, w1 and w2, we consider all 8 possible patterns that appear in a local context (5-word window), where we use wildcard (*) to indicate any token:

w1 w2
w1 * w2
w1 * * w2
w1 * * * w2
w2 w1
w2 * w1
w2 * * w1
w2 * * * w1

The relevance is then calculated by summing the logarithm of each of the 8 different counts. Finally, we compare the score of each candidate word and output the one with higher score.

4.2 Random Indexing

The relevance scores based on Random Indexing are provided by a tool FRanI (Higgins, 2004) and a model trained on the Touchstone Applied Science Associates (TASA) corpus which contains 750k sentences and covers diverse topics (from a diversity of textbooks up to the college level). Take the sentence at the beginning of this section for example, where only the words "a" and "friends" are related to the target word (either "complement" or "compliment") by either relevance measure. The relevance based on Random Indexing for (complement, friends) is 0.08, (compliment, friends) is 0.19 and both (compliment, a) and (complement, a) are 0 because "a" is in the stop word list. Meanwhile, the relevance based on first order word co-occurrence for (compliment, friends) is 7.39, (complement, friends) is 5.38, (compliment, a) is 13.25, and (complement, a) is 13.42. The system with either kind of relevance outputs "compliment".

4.3 System Combination

Since the numeric measurement of word co-occurrence is not as specific as the PAR features and less trustworthy, adding word co-occurrence information as features into the classifier along with n-gram counts, lexical and parse features will hurt the overall performance. It is more practical to combine the two approaches in the following fashion:

1. When the SVM classifier (using NG, LEX and PAR features) has high confidence (over a certain threshold) in the output label, output that label;

2. Otherwise, output the results of the word relatedness/co-occurrence-based system.

5 Evaluation

We evaluate the effectiveness of syntactic and distributional information on spelling correction. The performance of the system is measured by accuracy: the percentage of sentences in the test data for which the system chooses the correct word. We compare our results against two baselines: 1) MAJOR chooses the most frequent candidate from the

confusion set in the training corpus, and 2) Bergsma et al.'s (2010) best systems, NG+LEX. We include inflectional variants ("-ing", "-ed", "-s", "-ly") of confusion words in the evaluation, such as complementing, complimenting in addition to complement, compliment, because this better corresponds to the range of errors that may be encountered in actual use and thus increases the scope of the system as a real world application. Also following Bergsma et al. (2010), we use a linear SVM, more exactly, the L2-regularized L2-loss dual SVM in LIBLINEAR (Fan et al., 2008). Unlike Bergsma et al., who used development data to optimize parameters, we always use default parameters, since training data is limited for many of the words we are dealing with.

5.1 Data

Following Bergsma et al. (2009; 2010), the test examples are extracted from The New York Times (NYT) portion of Gigaword[1], but constrained to a 9-month publication time frame from October 2005 to July 2006. Unlike Bergsma et al. who use the same source as training data for the lexical features, our training data (for both lexical and parse features) comes from larger and more diverse news sources. We use the very large database from Sekine's n-gram search engine (Sekine, 2008) as training data, which consists of 1.9B words of newspaper text spanning 89 years from NYT, BBC, WSJ, Xinhua, etc.

We evaluate our systems on 5 confusion sets from Bergsma et al. (2009; 2010) and 9 commonly confused word pairs with moderate frequency in daily usage (randomly selected from those listed in English educational resources[2]). Shown in Table 2, these 9 sets of words appear much less frequently than the words selected by Bergsma et al., even given the fact that we are using a considerably large training corpus.

For each confusable word pair, sentences that contain either of the words are extracted to form training and test data. The word that appears in the original sentences of the news article is treated as the gold standard. For frequently occurring confusion word sets used by Bergsma et al., we extract up to 10k examples for testing, and up to 100k examples for training. For the 9 less frequent confusion word sets, we extract all the unique examples for training and testing from the above sources. The spelling correction system is evaluated by measuring its accuracy in comparison to the gold standard in test data. The error rate is the complement of accuracy.

[1]Available from the LDC as LDC2003T05

[2]Such as an English learning blog post at http://elisaenglish.pixnet.net/blog/post/1335194

Word Confusion Set	# in Training Corpus
adverse / averse	13.5k / 1.8k
advice / advise	62.k / 12.9k
allusion / illusion	1.0k / 5.4k
complement / compliment	6.8k / 3.1k
confidant / confident	2.4k / 63.6k
desert / dessert	24.7k / 3.7k
discreet / discrete	0.7k / 2.4k
elicit / illicit	1.9k / 10.0k
stationary / stationery	2.5k/2.3k
wander / wonder	3.3k / 39.5k

Table 2: Training Data Sizes for Common ESL Confused Words

Following Carlson et al. (2007) and Bergsma et al. (2009; 2010), we obtain the n-gram counts from the Google Web 1T 5-gram Corpus (Brants and Franz, 2006).

5.2 Experimental Results

We present the results for each set separately because each set may behave very differently, depending upon its frequency, part-of-speech, number of senses and other differences between the words in each confusion set. The overall accuracy across confusion sets is also presented to show the effectiveness of different approaches. The results are tested for statistical significance using McNemar's test of correlated proportions. The performance differences are marked as significant when $p < 0.05$.

5.2.1 Effectiveness of Parse Features

We exploit the n-gram counts (NG), lexical features (LEX) of Bergsma et al. (2010) and our own parse features (PAR) in linear SVM models.

The first comparison is between the supervised learning systems with LEX and LEX+PAR. As shown in Table 3, by exploiting our unique parse features, for the total 14 confusion sets, the accuracy increases on 12 sets and decreases on 2 sets. Overall, the spelling correction accuracy improves an ab-

solute 1.35% for our 9 confusion sets and 0.60% for Bergsma et al.'s 5 confusion sets.

The second comparison is to see how parse features interact with n-gram count features in a supervised classifier. The best system from (Bergsma et al., 2010) is listed in the table as "NG+LEX". As shown in Table 3, the parse features proved to be beneficial when augmenting this baseline, except for the decrease in accuracy on adverse, averse by only 2 cases out of 368, and among, between by 2 cases out of 10227. For all other confusion sets, parse features decrease the error rate by as much as 2.74% (absolute) and as much as 38.5% (relative). Improvements are statistically significant on all confusion sets together, although for each separate set, improvements are significant on only 5 sets, in part due to an insufficient number of test cases.

The reason that parse features are occasionally not helpful is because they sometimes include an uncommon word in dependencies, which happens to appear once with the wrong word but not with the correct word in the training data; or they sometimes include too common words, which bias the classifier in favor of the more frequent word in the confusion set. We also noticed that lexical features are not always helpful when added to n-gram count features, even for in-domain applications (i.e., with training data and test data coming from the same domain or corpus), as marked by underlines. However, lexical and parse features together show more significant and constant improvement over n-gram count-based models, as marked by α.

Of the six systems, every system that uses parse features gets the example correct in Section 1, "complementing the president"; LEX by itself also gets the example correct, but NG and NG+LEX fail.

In summary, our system NG+LEX+PAR outperforms the state-of-the-art system NG+LEX. It reduces the error rate by 12.4% across our 9 confusion sets and by 8.4% across Bergsma et al.'s 5 confusion sets. Both improvements are significant ($p < 0.05$) by the McNemar test. In addition, while NG+LEX is not always better than NG, NG+LEX+PAR is consistently better than NG.

5.2.2 Impact of Word Co-occurrence

The LIBLINEAR tool does not provide probability estimates for SVM models but Logistic Regression can. In this set of experiments, we train a Logistic Regression model with NG+LEX+PAR features and empirically set the confidence threshold at 0.6, as described in Section 4, based on the performance on two word pairs. In the combined system, when the Logistic Regression model estimates a probability higher than the threshold we output its results, otherwise we output the result of the system based on word co-occurrence.

Surprisingly, although Random Indexing takes into account more information than first-order word co-occurrence, it lowered overall performance substantially. Thus in Table 4, we only present results of using first-order word co-occurrence rather than Random Indexing. For all 12 confusion sets, distributional word co-occurrence information improves 9 sets and hurts 5 sets. Overall, it reduces the error rate slightly by 0.2% for our 9 sets and 1.5% for Bergsma et al.'s sets.

We believe there are two reasons why Random Indexing fared worse than first-order word co-occurrence: 1) Random Indexing considers co-occurrence on a document level, while our first-order word co-occurrence is limited to a 5-word window context. The latter is more suitable to context-sensitive spelling correction. 2) The model for Random Indexing is trained on a relatively small size corpus compared to the web-scale data we used to get n-gram count features for the classifier and thus is not able to introduce much new evidence besides the information carried by NG+LEX+PAR features.

Reason 2) also suggests why first-order co-occurrence helps on some occasions while not on other occasions. Its impact is limited because the word co-occurrence information overlaps with some of the PAR feature values as mentioned earlier. It improves some cases because it provides some new evidence from web-scale data to the system based on NG+LEX+PAR features. It introduces new errors because it simply favors the word that co-occurred more often regardless of other factors. Its impact is also limited because it is only considered when classifiers with NG+LEX+PAR features are not confident.

CONFUSION SET	# TEST	MAJOR	LEX	LEX+PAR	NG	NG+LEX	NG+LEX+PAR (&)
9 commonly cited ESL confusion pairs							
adverse / averse	368	85.87	**97.01**	96.74	91.03	**97.55**	97.01 (+22.2%) α
allusion / illusion	535	76.64	91.22	**91.40**	91.40	92.52	**93.08 (-7.5%)** α
complement / compliment	860	51.51	83.84	**85.12**	88.49	88.37	**89.53 (-10.0%)**
confidant / confident	2416	94.41	97.97	**98.30**	98.51	99.05	**99.09 (-4.3%)** α
desert / dessert	2357	70.81	90.71	**91.56**	87.31	93.68	**94.57 (-14.1%)** α*
discreet / discrete	219	79.45	84.48	**85.84**	85.84	90.41	**91.32 (-9.5%)** α
elicit / illicit	563	53.46	82.77	**95.56**	97.51	97.51	**98.22 (-28.6%)**
stationary / stationery	182	62.64	87.36	**92.31***	93.96	92.86	**95.60 (-38.5%)**
wander / wonder	6506	86.37	96.42	**97.42***	97.56	98.23	**98.48 (-13.9%)** α*
Total	13972	81.08	93.94	**95.29***	94.82	96.56	**96.99 (-12.4%)** α*
5 Original Bergsma pairs							
# among / between	10227	57.46	**91.89**	91.86	88.34	**93.60**	93.58 (+3.1%) α
# amount / number	7398	76.44	92.34	**93.16***	93.03	93.42	**94.08 (-10.1%)** α*
# cite / site	10185	95.71	99.42	**99.53**	99.16	99.52	**99.63 (-22.4%)**α
# peace / piece	7330	56.81	95.01	**97.01***	95.55	96.74	**97.46 (-22.2%)**α *
# raise / rise	9464	55.98	96.12	**96.64***	94.45	96.68	**97.05 (-11.5%)** α
Total	44604	68.92	95.09	**95.69***	94.07	96.09	96.42 (-8.4%) α

Table 3: Spelling correction precision (%), impact of adding parse features
SVM trained on 1G words of news text, tested on 9-months of NYT data.
*: Improvement of (NG+)LEX+PAR vs. (NG+)LEX is statistically significant.
α: Improvement of NG+LEX+PAR vs. NG is statistically significant.
&: Relative increase or decrease of error rate compared to "NG+LEX"
#: As in Bergsma et al. (2009; 2010) no morphological variants of the words are used in evaluation

CONFUSION SET	# TEST	MAJOR	CLASSIFIER	COMBINED SYSTEM (&)
9 commonly cited ESL confusion pairs				
adverse / averse	368	85.87	**97.55**	96.74 (+33.3%)
allusion / illusion	535	76.64	**92.34**	**92.34 (- 0.0%)**
complement / compliment	860	51.51	89.88	**90.81 (-9.2%)**
confidant / confident	2416	94.41	**99.13**	99.05 (+9.5%)
desert / dessert	2357	70.81	93.98	**94.23 (-3.7%)**
discreet / discrete	219	79.45	90.41	**91.78 (-14.3%)**
elicit / illicit	563	53.46	98.40	**98.76 (-22.2%)**
stationary / stationery	182	62.64	93.41	**93.96 (-9.1%)**
wander / wonder	6506	86.37	**98.49**	98.36 (+9.2%)
5 Original Bergsma pairs				
# among / between	10227	57.46	92.73	**92.73 (-0.1%)**
# amount / number	7398	76.44	93.44	**93.76 (-4.74%)**
# cite / site	10185	95.71	**99.49**	99.47 (+3.8%)
# peace / piece	7330	56.81	96.19	**96.38 (-5.0%)**
# raise / rise	9464	55.98	**96.66**	96.59 (+2.2%)

Table 4: Spelling correction accuracy (%), impact of combining word co-occurrence
CLASSIFIER: Logistic Regression trained on 1G words of news text, tested on 9-months NYT data.
COMBINED SYSTEM: CLASSIFER plus system based on first-order word co-occurrence.
&: Relative increase or decrease in error rate compared to CLASSIFIER
#: As in Bergsma et al. (2009; 2010), no morphological variants of the words are used in evaluation

6 Conclusions

We propose a novel approach that uses parse features and lexical features together to improve the performance of web-scale n-gram models for spelling correction. This method is especially adaptive when less training data are available, which is the case for confusable words that are not very frequently used. We also investigate the effectiveness of incorporating web-scale word co-occurrence and corpus-based semantic word relatedness (Random Indexing).

For future work, we will investigate using semantic information (e.g. WordNet) to extend n-gram models. It will be interesting to see if the usage of the word "compliment" in "complimenting the president" can be estimated by considering similar usages in the corpus, such as "complimenting the student" or by creating an n-gram database of synset patterns. We will investigate extending, to other applications, this general methodology combining distributional, semantic and syntactic information with language models.

Acknowledgments

We wish to thank Michael Flor of Educational Testing Service for his TrendStream tool, which provides fast access and easy manipulation of the Google N-gram Corpus. We also thank Derrick Higgins of Educational Testing Service for his Random Indexing support. We also thank Satoshi Sekine of New York University, Matthew Snover of City University of New York, and Jing Jiang of Singapore Management University for their advice.

References

Shane Bergsma, Dekang Lin, and Randy Goebel. 2009. Web-scale n-gram models for lexical disambiguation. In *IJCAI*.

Shane Bergsma, Emily Pitler, and Dekang Lin. 2010. Creating robust supervised classifiers via web-scale n-gram data. In *ACL*.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Available at http://www.ldc.upenn.edu.

Alexander Budanitsky and Graeme Hirst. 2001. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *ACL Workshop on WordNet and Other Lexical Resources*.

Andrew Carlson and Ian Fette. 2007. Memory-based context sensitive spelling correction at web scale. In *ICMLA*.

Martin Chodorow, Joel Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, pages 25–30.

Silviu Cucerzan and David Yarowsky. 2002. Augmented mixture models for lexical disambigua-tion. In *EMNLP*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, Genoa, Italy.

Scott Deerwester, Susan Dumais, George Furmas, Thomas Landauer, and Richar Harshman. 1990. Indexing by latent semantic analysis. *The American Society for Information Science*.

Dmitriy Dligach and Martha Palmer. 2008. Novel semantic features for verb sense disambiguation. In *ACL*.

Philip Edmonds. 1997. Choosing the word most typical in context using a lexical co-occurrence network. In *EACL*.

Mohammed Ali Elmi and Martha Evans. 1998. Spelling correction using context. In *COLING*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Machine Learning Research*, 9(1871-1874).

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of COLING*, Manchester, UK.

Rachele De Felice and Stephen G. Pulman. 2009. Automatic detection of preposition errors in learner writing. *CALICO Journal*, 26(3).

Michael Gamon, Jianfeng Gao, Chris Brockett, Alex Klementiev, William B. Dolan, Dmitriy Belenko, and Lucy Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 449–456, Hyderabad, India.

Andrew Golding and Dan Roth. 1996. Applying Winnow to context-sensitive spelling correction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 182–190.

Andrew Golding and Dan Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

Andrew Golding. 1995. A Bayesian hybrid method for context sensitive spelling correction. In *Proceedings*

of the Third Workshop on Very Large Corpora (WVLC-3), pages 39–53.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Matthieu Hermet, Alain Désilets, and Stan Szpakowicz. 2008. Using the web as a linguistic resource to automatically correct lexico-syntactic errors. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*, pages 390–396, Marrekech, Morocco.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *Companion Volume to the Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 145–148.

Michael Jones and James Martin. 1997. Contextual spelling correction using latent semantic analysis. In *ANLC*.

Thomas Landauer, Darrell Laham, and Peter Foltz. 1998. Learning human-like knowledge by singular value decomposition: A progress report. *Advances in Neural Information Processing Systems*, 10:45–51.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 21:1–31.

John Lee and Ola Knutsson. 2008. The role of pp attachment in preposition generation. In *CICLING*.

Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In *ICML*.

Saif Mohammad and Graeme Hist. 2006. Distributional measures of concept distance: A task-oriented evaluation. In *EMNLP*.

Alla Rozovskaya and Dan Roth. 2010. Training paradigms for correcting errors in grammar and usage. In *ACL*.

Magnus Sahlgren. 2006. *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis.

Joel Tetreault and Martin Chodorow. 2008. The ups and downs of prepostion error detection in esl writing. In *COLING*.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In *ACL*.

Casey Whitelaw, Ben Hutchinson, Grace Y. Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *ACL*.

Discriminating Gender on Twitter

John D. Burger and **John Henderson** and **George Kim** and **Guido Zarrella**
The MITRE Corporation
202 Burlington Road
Bedford, Massachusetts, USA 01730
{john,jhndrsn,gkim,jzarrella}@mitre.org

Abstract

Accurate prediction of demographic attributes from social media and other informal online content is valuable for marketing, personalization, and legal investigation. This paper describes the construction of a large, multilingual dataset labeled with gender, and investigates statistical models for determining the gender of uncharacterized Twitter users. We explore several different classifier types on this dataset. We show the degree to which classifier accuracy varies based on tweet volumes as well as when various kinds of profile metadata are included in the models. We also perform a large-scale human assessment using Amazon Mechanical Turk. Our methods significantly out-perform both baseline models and almost all humans on the same task.

1 Introduction

The rapid growth of social media in recent years, exemplified by Facebook and Twitter, has led to a massive volume of user-generated informal text. This in turn has sparked a great deal of research interest in aspects of social media, including automatically identifying latent demographic features of online users. Many latent features have been explored, but gender and age have generated great interest (Schler et al., 2006; Burger and Henderson, 2006; Argamon et al., 2007; Mukherjee and Liu, 2010; Rao et al., 2010). Accurate prediction of these features would be useful for marketing and personalization concerns, as well as for legal investigation.

In this work, we investigate the development of high-performance classifiers for identifying the gender of Twitter users. We cast gender identification as the obvious binary classification problem, and explore the use of a number of text-based features. In Section 2, we describe our Twitter corpus, and our methods for labeling a large subset of this data for gender. In Section 3 we discuss the features that are used in our classifiers. We describe our Experiments in Section 4, including our exploration of several different classifier types. In Section 5 we present and analyze performance results, and discuss some directions for acquiring additional data by simple self-training techniques. Finally in Section 6 we summarize our findings, and describe extensions to the work that we are currently exploring.

2 Data

Twitter is a social networking and micro-blogging platform whose users publish short messages or *tweets*. In late 2010, it was estimated that Twitter had 175 million registered users worldwide, producing 65 million tweets per day (Miller, 2010). Twitter is an attractive venue for research into social media because of its large volume, diverse and multilingual population, and the generous nature of its Terms of Service. This has led many researchers to build corpora of Twitter data (Petrovic et al., 2010; Eisenstein et al., 2010). In April 2009, we began sampling data from Twitter using their API at a rate of approximately 400,000 tweets per day. This represented approximately 2% of Twitter's daily volume at the time, but this fraction has steadily decreased to less than 1% by 2011. This decrease is because we sample roughly the same number of tweets every day while Twitter's overall volume has increased markedly. Our corpus thus far contains approximately 213 million tweets from 18.5 million users, in many different languages.

In addition to the tweets that they produce, each Twitter user has a profile with the following free-text fields:

- Screen name (e.g., *jsmith92*, *kingofpittsburgh*)
- Full name (e.g., *John Smith*, *King of Pittsburgh*)
- Location (e.g., *Earth*, *Paris*)
- URL (e.g., the user's web site, Facebook page, etc.)
- Description (e.g., *Retired accountant and grandfather*)

All of these except screen name are completely optional, and all may be changed at any time. Note that none

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1301–1309,
Edinburgh, Scotland, UK, July 27–31, 2011.

	Users	Tweets
Training	146,925	3,280,532
Development	18,380	403,830
Test	18,424	418,072

Figure 1: Dataset Sizes

of the demographic attributes we might be interested in are present, such as gender or age. Thus, the existing profile elements are not directly useful when we wish to apply supervised learning approaches to classify tweets for these target attributes. Other researchers have solved this problem by using labor-intensive methods. For example, Rao et al. (2010) use a focused search methodology followed by manual annotation to produce a dataset of 500 English users labeled with gender. It is infeasible to build a large multilingual dataset in this way, however.

Previous research into gender variation in online discourse (Herring et al., 2004; Huffaker, 2004) has found it convenient to examine blogs, in part because blog sites often have rich profile pages, with explicit entries for gender and other attributes of interest. Many Twitter users use the URL field in their profile to link to another facet of their online presence. A significant number of users link to blogging websites, and many of these have well-structured profile pages indicating our target attributes. In many cases, these are not free text fields. Users on these sites must select gender and other attributes from drop-down menus in order to populate their profile information. Accordingly, we automatically followed the Twitter URL links to several of the most represented blog sites in our dataset, and sampled the corresponding profiles. By attributing this blogger profile information to the associated Twitter account, we created a corpus of approximately 184,000 Twitter users labeled with gender.

We partitioned our dataset by user into three distinct subsets, training, development, and test, with sizes as indicated in Figure 1. That is, all the tweets from each user are in a single one of the three subsets. This is the corpus we use in the remainder of this paper.

This method of gleaning supervised labels for our Twitter data is only useful if the blog profiles are in turn accurate. We conducted a small-scale quality assurance study of these labels. We randomly selected 1000 Twitter users from our training set and manually examined the description field for obvious indicators of gender, e.g., *mother to 3 boys* or *just a dude*. Only 150 descriptions (15% of the sample) had such an explicit gender cue. 136 of these also had a blog profile with the gender selected, and in all of these the gender cue from the user's Twitter description agreed with the corresponding blog profile. This may only indicate that people who misrepresent their gender are simply consistent across different aspects of their online presence. However, the effort involved in maintaining this deception in two different places suggests that the blog labels on the Twitter data are largely reliable.

Initial analysis using the blog-derived labels showed that our corpus is composed of 55% females and 45% males. This is consistent with the results of an earlier study which used name/gender correlations to estimate that Twitter is 55% female (Heil and Piskorski, 2009). Figure 2 shows several statistics broken down by gender, including the Twitter users who did not indicate their gender on their blog profile. In our dataset females tweet at a higher rate than males and in general users who provide their gender on their blog profile produce more tweets than users who do not. Additionally, of the 150 users who provided a gender cue in their Twitter user description, 105 were female (70%). Thus, females appear more likely to provide explicit indicators about their gender in our corpus.

The average number of tweets per user is 22 and is fairly consistent across our traing/dev/test splits. There is wide variance, however, with some users represented by only a single tweet, while the most prolific user in our sample has nearly 4000 tweets.

It is worth noting that many Twitter users do not tweet in English. Table 3 presents an estimated breakdown of language use in our dataset. We ran automatic language ID on the concatenated tweet texts of each user in the training set. The strong preponderance of English in our dataset departs somewhat from recent studies of Twitter language use (Wauters, 2010). This is likely due in part to sampling methodology differences between the two studies. The subset of Twitter users who also use a blog site may be different from the Twitter population as a whole, and may also be different from the users tweeting during the three days of Wauters's study. There are also possible longitudinal differences: English was the dominant language on Twitter when the online service began in 2006, and this was still the case when we began sampling tweets in 2009, but the proportion of English tweets had steadily dropped to about 50% in late 2010. Note that we do not use any explicit encoding of language information in any of the experiments described below.

Our Twitter-blog dataset may not be entirely representative of the Twitter population at general, but this has at least one advantage. As with any part of the Internet, spam is endemic to Twitter. However by sampling only Twitter users with blogs we have largely filtered out spammers from our dataset. Informal inspection of a few thousand tweets revealed a negligible number of commercial tweets.

3 Features

Tweets are tagged with many sources of potentially discriminative metadata, including timestamps, user color

	Users		Tweets		Mean tweets
	Count	Percentage	Count	Percentage	per user
Female	100,654	42.3%	2,429,621	47.7%	24.1
Male	83,075	35.0	1,672,813	32.8	20.1
Not provided	53,817	22.7	993,671	19.5	18.5

Figure 2: Gender distribution in our blog-Twitter dataset

Language	Users	Percentage
English	98,004	66.7%
Portuguese	21,103	14.4
Spanish	8,784	6.0
Indonesian	6,490	4.4
Malay	1,401	1.0
German	1,220	0.8
Chinese	985	0.7
Japanese	962	0.7
French	878	0.6
Dutch	761	0.5
Swedish	686	0.5
Filipino	643	0.4
Italian	631	0.4
Other	4,377	3.0

Figure 3: Language ID statistics from training set

preferences, icons, and images. We have restricted our experiments to a subset of the textual sources of features as listed in Figure 4.

We use the content of the tweet text as well as three fields from the Twitter user profile described in Section 2: full name, screen name, and description. For each user in our dataset, a field is in general a *set* of text strings. This is obviously true for tweet texts but is also the case for the profile-based fields since a Twitter user may change any part of their profile at any time. Because our sample spans points in time where users have changed their screen name, full name or description, we include all of the different values for those fields as a set. In addition, a user may leave their description and full name blank, which corresponds to the empty set.

In general, our features are quite simple. Both word- and character-level ngrams from each of the four fields are included, with and without case-folding. Our feature functions do not count multiple occurrences of the same ngram. Initial experiments with count-valued feature functions showed no appreciable difference in performance. Each feature is a simple Boolean indicator representing presence or absence of the word or character ngram in the set of text strings associated with the particular field. The extracted set of such features represents the item to the classifier.

For word ngrams, we perform a simple tokenization

	Feature extraction		
	Char ngrams	Word ngrams	Distinct features
Screen name	1–5	*none*	432,606
Full name	1–5	1	432,820
Description	1–5	1–2	1,299,556
Tweets	1–5	1–2	13,407,571
Total			15,572,522

Figure 4: Feature types and counts

that separates words at transitions between alphanumeric characters and non-alphanumeric.[1] We make no attempt to tokenize unsegmented languages such as Chinese, nor do we perform morphological analysis on language such as Korean; we do no language-specific processing at all. We expect the character-level ngrams to extract useful information in the case of such languages.

Figure 4 indicates the details and feature counts for the fields from our training data. We ignore all features exhibited by fewer than three users.

4 Experiments

We formulate gender labeling as the obvious binary classification problem. The sheer volume of data presents a challenge for many of the available machine learning toolkits, e.g. WEKA (Hall et al., 2009) or MALLET (McCallum, 2002). Our 4.1 million tweet training corpus contains 15.6 million distinct features, with feature vectors for some experiments requiring over 20 gigabytes of storage. To speed experimentation and reduce the memory footprint, we perform a one-time feature generation preprocessing step in which we convert each feature pattern (such as "caseful screen name character trigram: Joh") to an integer codeword. The learning algorithms do not access the codebook at any time and instead deal solely with vectors of integers. We compress the data further by concatenating all of a user's features into a single vector that represents the union of every tweet produced by that user. This condenses the dataset to about 180,000 vectors occupying 11 gigabytes of storage.

We performed initial feasibility experiments using a wide variety of different classifier types, including Support Vector Machines, Naive Bayes, and Balanced Win-

[1] We use the standard regular expression pattern `\b`.

now2 (Littlestone, 1988). These initial experiments were based only on caseful word unigram features from tweet texts, which represent less than 3% of the total feature space but still include large numbers of irrelevant features. Performance as measured on the development set ranged from Naive Bayes at 67.0% accuracy to Balanced Winnow2 at 74.0% accuracy. A LIBSVM (Chang and Lin, 2001) implementation of SVM with a linear kernel achieved 71.8% accuracy, but required over fifteen hours of training time while Winnow needed less than seven minutes. No classifier that we evaluated was able to match Winnow's combination of accuracy, speed, and robustness to increasing amounts of irrelevant features.

We built our own implementation of the Balanced Winnow2 algorithm which allowed us to iterate repeatedly over the training data on disk rather than caching the entire dataset in memory. This reduced our memory requirements to the point that we were able to train on the entire dataset using a single machine with 8 gigabytes of RAM.

We performed a grid search to select learning parameters by measuring their affect on Winnow's performance on the development set. We found that two sets of parameters were required: a low learning rate (0.03) was effective when using only one type of input feature (such as only screen name features, or only tweet text features), and a higher learning rate (0.20) was required when mixing multiple types of features in one classifier. In both cases we used a relatively large margin (35%) and cooled the learning rate by 50% after each iteration.

These learning parameters were used during all of the experiments that follow. All gender prediction models were trained using data from the training set and evaluated on data from the development set. The test set was held out entirely until we finalized our best performing models.

4.1 Field combinations

We performed a number of experiments with the Winnow algorithm described above. We trained it on the training set and evaluated on the development set for each of the four user fields in isolation, as well as various combinations, in order to simulate different use cases for systems that perform gender prediction from social media sources. In some cases we may have all of the metadata fields available above, while in other cases we may only have a sample of a user's tweet content or perhaps just one tweet. We simulated the latter condition by randomly selecting a single tweet for each dev and test user; this tweet was used for all evaluations of that user under the single-tweet condition. Note, however, that for training the single tweet classifier, we do not concatenate all of a user's tweets as described above. Instead, we pair each user in the training set with each of their tweets in turn, in order to take advantage of all the training data. This amounted to over 3 million training instances for the single tweet condition.

We paid special attention to three conditions: single tweet, all fields, and all tweets. For these conditions, we evaluated the learned models on the training data, the development set, and the test set, to study over-training and generalization. Note that for all experiments, the evaluation includes some users who have left their full name or description fields blank in their profile.

In all cases, we compare results to a maximum likelihood baseline that simply labels all users female.

4.2 Human performance

We wished to compare our classifier's efficacy to human performance on the same task. A number of researchers have recently experimented with the use of Amazon Mechanical Turk (AMT) to create and evaluate human language data (Callison-Burch and Dredze, 2010). AMT and other crowd-sourcing platforms allow simple tasks to be posted online for large numbers of anonymous workers to complete.

We used AMT to measure human performance on gender determination for the all tweets condition. Each AMT worker was presented with all of the tweet texts from a single Twitter user in our development set and asked whether the author was male or female. We redundantly assigned five workers to each Twitter user, for a total of 91,900 responses from 794 different workers. We experimented with a number of ways to combine the five human labels for each item, including a simple majority vote and a more sophisticated scheme using an expectation maximization algorithm.

4.3 Self-training

Our final experiments were focused on exploring the use of unlabeled data, of which we have a great deal. We performed some initial experiments on a self-training approach to labeling more data. We trained the all-fields classifier on half of our training data, and applied it to the other half. We trained a new classifier on this full training set, which now included label errors introduced by the limitations of the first classifier. This provided a simulation of a self-training setup using half the training data. Any robust gains due to self-training should be revealed by this setup.

5 Results

5.1 Field combinations

Figure 5 shows development set performance on various combinations of the user fields, all of which outperform the maximum likelihood baseline that classifies all users as female. The single most informative field with respect

Baseline (F)	54.9%
One tweet text	67.8
Description	71.2
All tweet texts	75.5
Screen name (e.g. *jsmith92*)	77.1
Full name (e.g. *John Smith*)	89.1
Tweet texts + screen name	81.4
Tweet texts + screen name + description	84.3
All four fields	92.0

Figure 5: Development set accuracy using various fields

Condition	Train	Dev	Test
Baseline (F)	54.8%	54.9	54.3
One tweet text	77.8	67.8	66.5
Tweet texts	77.9	75.5	74.5
All fields	98.6	92.0	91.8

Figure 6: Accuracy on the training, development and test sets

to gender is the user's full name, which provides an accuracy of 89.1%. Screen name is often a derivative of full name, and it too is informative (77.1%), as is the user's self-assigned description (71.2).

Using only tweet texts performs better than using only the user description (75.5% vs. 71.2). Tweet texts are sufficient to decrease the error by nearly half over the all-female prior. It appears that the tweet texts convey more about a Twitter user's gender than their own self-descriptions. Even a single (randomly selected) tweet text contains some gender-indicative information (67.2%). These results are similar to previous work. Rao et al. (2010) report results of 68.7% accuracy on gender from tweet texts alone using an ngram-only model, rising to 72.3 with hand-crafted "sociolinguistic-based" features. Test set differences aside, this is comparable with the "All tweet texts" line in Figure 5, where we achieve an accuracy of 75.5%.

Performance of models built from various aggregates of the four basic fields are shown in Figure 5 as well. The combination of tweet texts and a screen name represents a use case common to many different social media sites, such as chat rooms and news article comment streams. The performance of this combination (81.4%) is significantly higher than either of the individual components. As we have observed, full name is the single most informative field. It out-performs the combination of the other three fields, which perform at 84.3%. Finally, the classifier that has access to features from all four fields is able to achieve an accuracy of 92.0%.

The final test set accuracy is shown in Figure 6. This test set was held out entirely during development and has been evaluated only with the four final models reported

Rank	MI	Feature f	$P(Female\|f)$
1	0.0170	!	0.601
2	0.0164	_:	0.656
3	0.0163	_lov	0.687
4	0.0162	love	0.680
5	0.0161	lov	0.676
6	0.0160	_love	0.689
7	0.0160	!_	0.618
8	0.0149	:)	0.697
9	0.0148	y!	0.687
10	0.0145	**my**	0.637
11	0.0143	love_	0.691
12	0.0143	haha	0.705
13	0.0141	my_	0.634
14	0.0140	_my	0.637
15	0.0140	_:)	0.697
16	0.0139	_my	0.634
17	0.0138	!_i	0.711
18	0.0138	hah	0.698
19	0.0137	_hah	0.714
20	0.0135	_so	0.661
21	0.0134	_haha	0.714
22	0.0132	**so**	0.661
23	0.0128	_i	0.618
24	0.0127	ooo	0.708
25	0.0126	!_i	0.743
26	0.0123	i_lov	0.728
27	0.0120	ove_	0.671
28	0.0117	ay!	0.718
29	0.0116	aha	0.678
30	0.0116	<3	0.856
31	0.0115	_cute	0.826
32	0.0114	i_lo	0.704
33	0.0114	:)$	0.701
34	0.0110	:(	0.731
35	0.0109	_:)$	0.701
36	0.0109	!$	0.614
37	0.0107	ahah	0.716
38	0.0106	_<3	0.857
464	0.0051	_ht	♂ 0.506
465	0.0051	hank	0.641
466	0.0051	too_	0.659
467	0.0051	_yay!	0.818
468	0.0051	_http	♂ 0.506
469	0.0051	_htt	♂ 0.506
624	0.0047	Googl	♂ 0.317
625	0.0047	ing!_	0.718
626	0.0047	hair_	0.749
627	0.0047	_b	0.573
628	0.0047	y_:	0.725
629	0.0046	Goog	♂ 0.318

Figure 7: A selection of tweet text features, ranked by mutual information. Character ngrams in Courier, words in **bold**. Underscores are spaces, $ matches the end of the tweet text. ♂ marks "male" features.

in this figure. The difference between the scores on the train and development sets show how well the model can fit the data. There are features in the user name and user screen name fields that make the data trivially separable. The tweet texts, however, present more ambiguity for the learners. The difference between the development and test set scores suggest that only minimal hill-climbing occurred during our development.

We have performed experiments to better understand how performance scales with training data size. Figure 8 shows how performance increases for both the all-fields and tweet-texts-only classifiers as we train on more users, with little indication of leveling off.

As discussed in Section 2, there is wide variance in the number of tweets available from different users. In Figure 9 we show how the tweet text classifier's accuracy increases as the number of tweets from the user increases. Each point is the average classifier accuracy for the user cohort with exactly that many tweets in our dev set. Performance increases given more tweets, although the averages get noisy for the larger tweet sets, due to successively smaller cohort sizes.

Some of the most informative features from tweet texts are shown in Figure 7, ordered by mutual information with gender. There are far more of these strong features for the female category than the male: only five of the top 1000 features are associated more strongly with males, i.e. they have lower $P(Female|feature)$ than the prior, $P(Female) = 0.55$.

Some of these features are content-based (*hair*, and several fragments of *love*), while others are stylistic (*ooo*, several emoticons). The presence of *http* as a strong male feature might be taken to indicate that men include links in their tweet texts far more often than women, but a cursory examination seems to show instead that women are simply more likely to include "bare" links, e.g., `emnlp.org` vs. `http://emnlp.org`.

5.2 Human performance

Figure 10 shows the results of the human performance benchmarks using Amazon Mechanical Turk. The raw per-response performance is 60.4%, only moderately better than the all-female baseline. When averaged across workers, however, this improves substantially, to 68.7. This would seem to indicate that there were a few poor workers who did many annotations, and in fact when we limit the performance average to those workers who produced 100 or more responses, we do see a degradation to 62.2.

The problem of poor quality workers is endemic to anonymous crowd sourcing platforms like Mechanical Turk. A common way to combat this is to use redundancy, with a simple majority vote to choose among multiple responses for each item. This allows us to treat the five workers who responded to each item as an ensemble. As Figure 10 indicates, this provides some improvement over the raw result (65.7% vs. 60.4). A different approach, first proposed by Dawid and Skene (1979), is to use an expectation maximization algorithm to estimate the quality of each source of labels, as well as estimate the posterior for each item. In this case, the first is an AMT worker's capability and the second is the distribution of gender labels for each Twitter user.

Baseline	54.9
Average response	60.4
Average worker	68.7
Average worker (100 or more responses)	62.2
Worker ensemble, majority vote	65.7
Worker ensemble, EM-adjusted vote	67.3
Winnow all-tweet-texts classifier	75.5

Figure 10: Comparing with humans on the all tweet texts task

The Dawid and Skene approach has previously been applied to Mechanical Turk responses (Ipeirotis et al., 2010). We used their implementation on our AMT results but with only moderate improvement over the simple majority ensemble (67.3% vs. 65.7). All of the aggregate human results are substantially below the all-tweet-texts classifier score, suggesting that this is a difficult task for people to perform. As Figure 11 indicates, most workers perform below 80% accuracy, and less than 5% of the prolific workers out-perform the automatic classifier. These high-scoring workers may indeed be good at the task, or they may have simply been assigned a less-difficult subset of the data. Figure 12 illustrates this by showing aligned worker performance and classifier performance on the precise set of items that each worker performed on. Here we see that, with few exceptions, the automatic classifier performs as well or better than the AMT workers on their subset.

5.3 Self-training

Finally, as described in Section 4.3, we performed some initial experiments on a self-training approach to labeling more data. As described above the all-fields classifier achieves an accuracy of 92% on the development set when trained on the full training set. Training on half of the training data results in a drop to 91.1%. The second classifier trained on the full training set, but with some label errors introduced by the first, had further degraded performance of 90.9%. Apparently the errorful labels introduced by the simplistic self-training procedure overwhelmed any new information that might have been gained from the additional data. We are continuing to explore ways to use the large amounts of unsupervised data in our corpus.

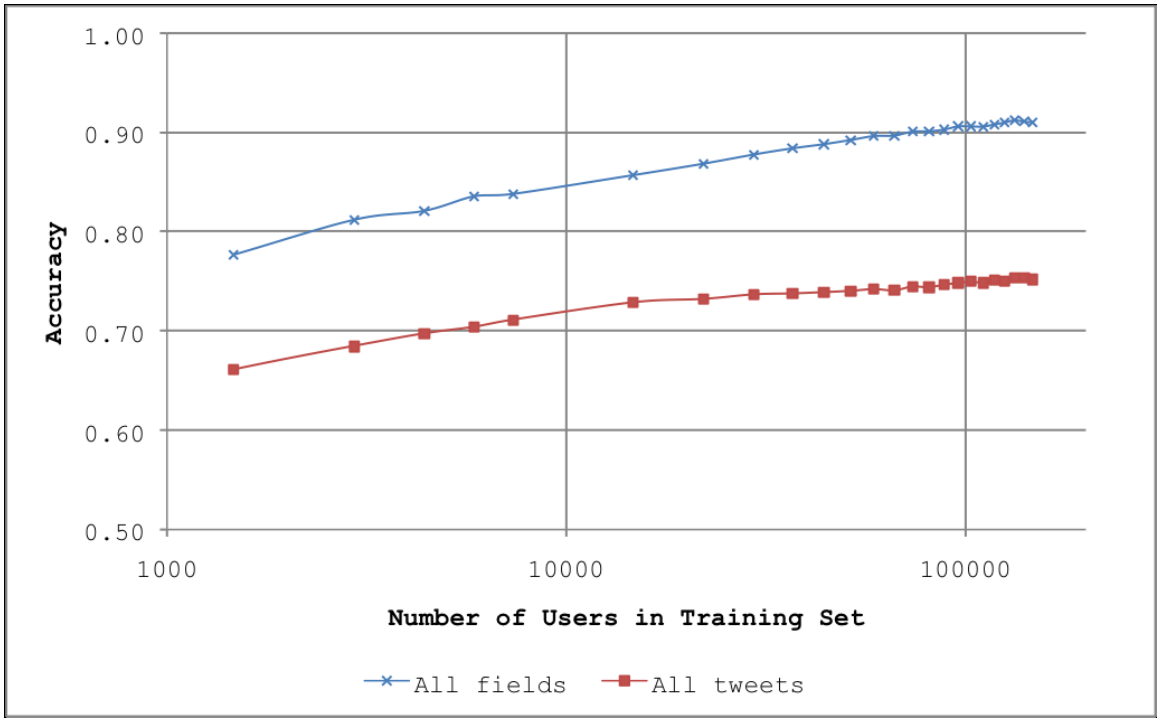


Figure 8: Performance increases when training with more users

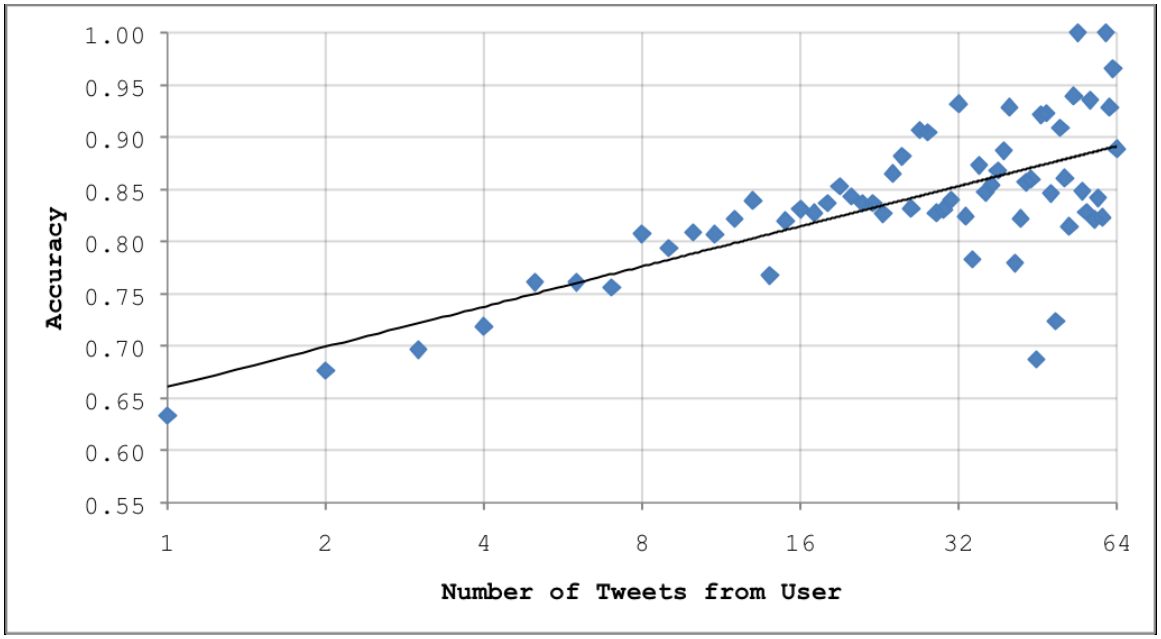


Figure 9: Performance increases with more tweets from target user

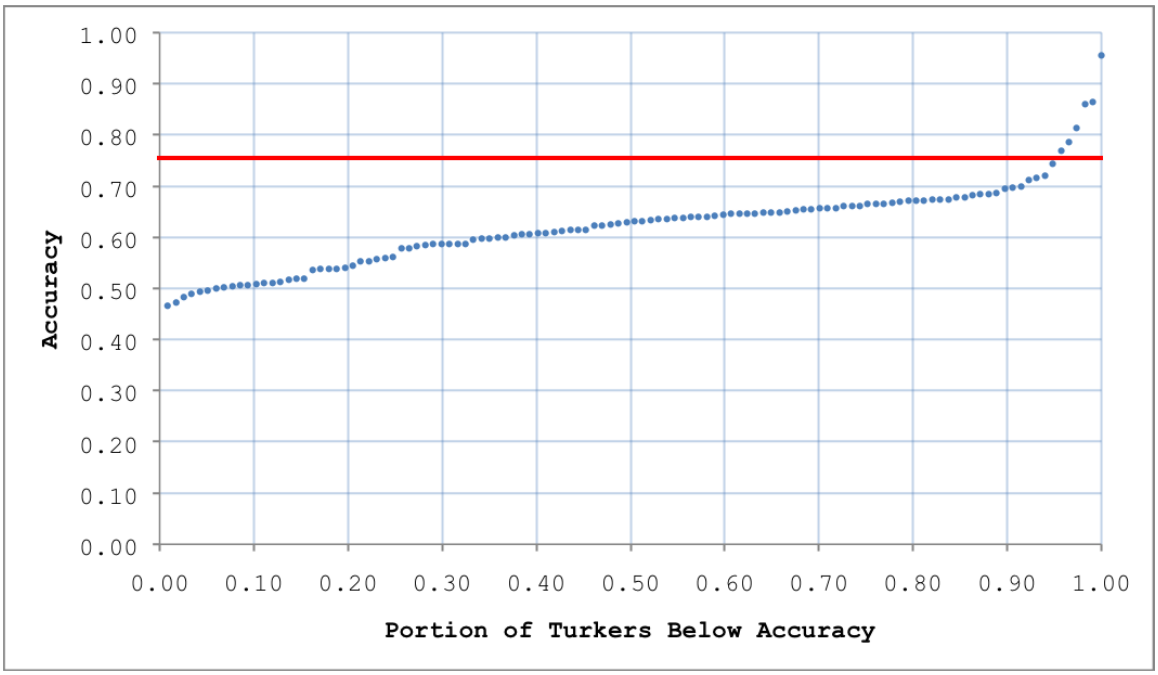


Figure 11: Human accuracy in rank order (100 responses or more), with classifier performance (line)

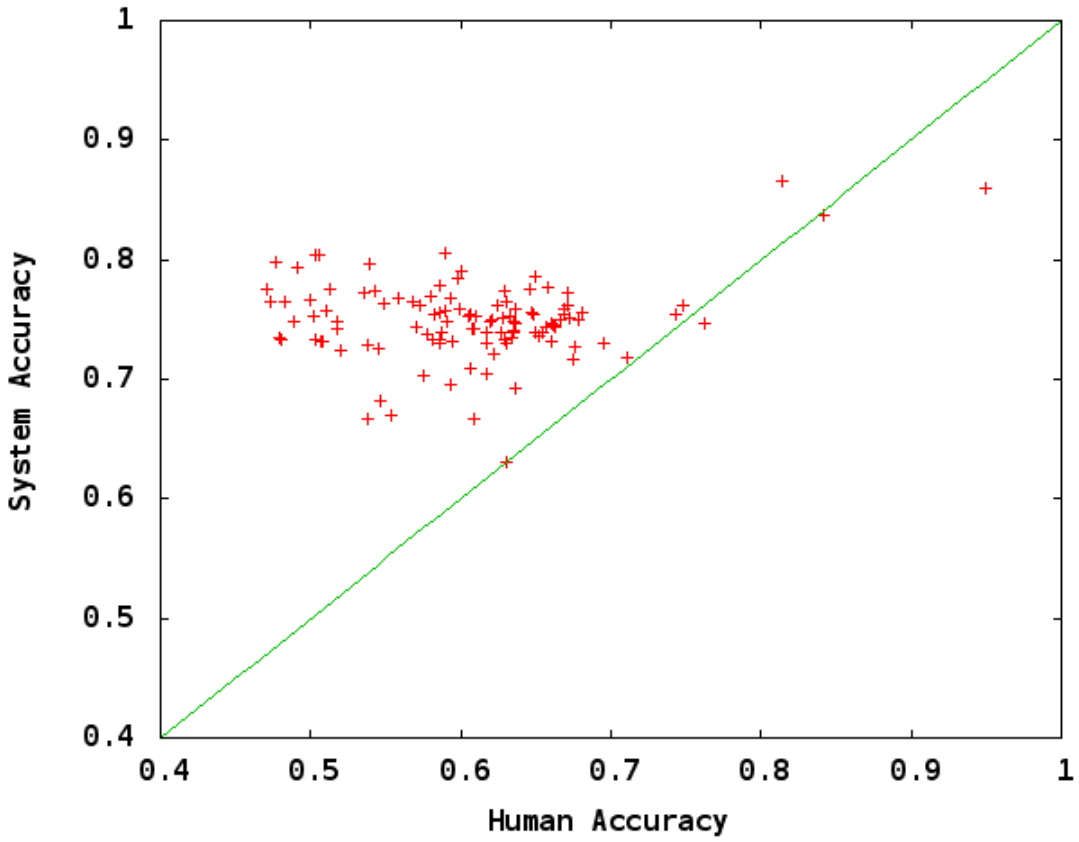


Figure 12: Classifier vs. human accuracy on the same subsets (100 responses or more)

6 Conclusion

In this paper, we have presented several configurations of a language-independent classifier for predicting the gender of Twitter users. The large dataset used for construction and evaluation of these classifiers was drawn from Twitter users who also completed blog profile pages.

These classifiers were tested on the largest set of gender-tagged tweets to date that we are aware of. The best classifier performed at 92% accuracy, and the classifier relying only on tweet texts performed at 76% accuracy. Human performance was assessed on this latter condition, and only 5% of 130 humans performed 100 or more classifications with higher accuracy than this machine.

In future work, we will explore how well such models carry over to gender identification in other informal online genres such as chat and forum comments. Furthermore, we have been able to assign demographic features beside gender, including age and location, to our Twitter dataset. We have begun to build classifiers for these features as well.

Acknowledgements

The authors would like to thank the anonymous reviewers. This work was funded under the MITRE Innovation Program.

References

Shlomo Argamon, Moshe Koppel, James W. Pennebaker, and Jonathan Schler. 2007. Mining the blogosphere: Age, gender, and the varieties of self-expression. *First Monday*, 12(9), September.

John D. Burger and John C. Henderson. 2006. An exploration of observable features related to blogger age. In *Computational Approaches to Analyzing Weblogs: Papers from the 2006 AAAI Spring Symposium*. AAAI Press.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

A.P. Dawid and A.M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1).

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Conference on Empirical Methods on Natural Language Processing*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1).

Bill Heil and Mikolaj Jan Piskorski. 2009. New Twitter research: Men follow men and nobody tweets. *Harvard Business Review*, June 1.

Susan C. Herring, Inna Kouper, Lois Ann Scheidt, and Elijah L. Wright. 2004. Women and children last: The discursive construction of weblogs. In L. Gurak, S. Antonijevic, L. Johnson, C. Ratliff, and J. Reyman, editors, *Into the Blogosphere: Rhetoric, Community, and Culture of Weblogs*. `http://blog.lib.umn.edu/blogosphere/`.

David Huffaker. 2004. Gender similarities and differences in online identity and language use among teenage bloggers. Master's thesis, Georgetown University. `http://cct.georgetown.edu/thesis/DavidHuffaker.pdf`.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on Amazon Mechanical Turk. In *Proceedings of the Second Human Computation Workshop (KDD-HCOMP 2010)*.

Nick Littlestone. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2, April.

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

Claire Cain Miller. 2010. Why Twitter's C.E.O. demoted himself. *New York Times*, October 30. `http://www.nytimes.com/2010/10/31/technology/31ev.html`.

Arjun Mukherjee and Bing Liu. 2010. Improving gender classification of blog authors. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October. Association for Computational Linguistics.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. The Edinburgh Twitter corpus. In *Computational Linguistics in a World of Social Media*. AAAI Press. Workshop at NAACL.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in Twitter. In *2nd International Workshop on Search and Mining User-Generated Content*. ACM.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. 2006. Effects of age and gender on blogging. In *Computational Approaches to Analyzing Weblogs: Papers from the 2006 AAAI Spring Symposium*. AAAI Press, March.

Robin Wauters. 2010. Only 50% of Twitter messages are in English, study says. *TechCrunch*, February 1. `http://techcrunch.com/2010/02/24/twitter-languages/`.

Identifying and Following Expert Investors in Stock Microblogs

[1]Roy Bar-Haim, [1]Elad Dinur, [1,2]Ronen Feldman, [1]Moshe Fresko and [1]Guy Goldstein
[1]Digital Trowel, Airport City, Israel
[2]School of Business Administration, The Hebrew University of Jerusalem, Jerusalem, Israel
{roy, moshe}@digitaltrowel.com, ronen.feldman@huji.ac.il

Abstract

Information published in online stock investment message boards, and more recently in stock microblogs, is considered highly valuable by many investors. Previous work focused on aggregation of sentiment from all users. However, in this work we show that it is beneficial to distinguish expert users from non-experts. We propose a general framework for identifying expert investors, and use it as a basis for several models that predict stock rise from stock microblogging messages (stock tweets). In particular, we present two methods that combine expert identification and per-user unsupervised learning. These methods were shown to achieve relatively high precision in predicting stock rise, and significantly outperform our baseline. In addition, our work provides an in-depth analysis of the content and potential usefulness of stock tweets.

1 Introduction

Online investment message boards such as Yahoo! Finance and Raging Bull allow investors to share trading ideas, advice and opinions on public companies. Recently, stock microblogging services such as StockTwits (which started as a filtering service over the Twitter platform) have become very popular. These forums are considered by many investors as highly valuable sources for making their trading decisions.

This work aims to mine useful investment information from messages published in stock microblogs. We shall henceforth refer to these messages as *stock tweets*. Ultimately, we would like to transform those tweets into buy and sell decisions. Given a set of stock-related messages, this process typically comprises two steps:

1. Classify each message as "bullish" (having a positive outlook on the stock), "bearish" (having a negative outlook on the stock), or neutral.
2. Make trading decisions based on these message classifications.

Previous work on stock investment forums and microblogs usually regarded the first step (message classification) as a sentiment analysis problem, and aligned *bullish* with positive sentiment and *bearish* with negative sentiment. Messages were classified by matching positive and negative terms from sentiment lexicons, learning from a hand-labeled set of messages, or some combination of the two (Das and Chen, 2007; Antweiler and Frank, 2004; Chua et al., 2009; Zhang and Skiena, 2010; Sprenger and Welpe, 2010). Trading decisions were made by aggregating the sentiment for a given stock over all the tweets, and picking stocks with strongest sentiment signal (buying the most bullish stocks and short-selling the most bearish ones).

Sentiment aggregation reflects the opinion of the investors community as a whole, but overlooks the variability in user expertise. Clearly, not all investors are born equal, and if we could tell experts from non-experts, we would reduce the noise in these forums and obtain high-quality signals to follow. This paper presents a framework for identifying experts in stock microblogs by monitoring their performance in a training period. We show that following the experts results in more precise predictions.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1310–1319,
Edinburgh, Scotland, UK, July 27–31, 2011.

Based on the expert identification framework, we experiment with different methods for deriving predictions from stock tweets. While previous work largely aligned bullishness with message sentiment, our in-depth content analysis of stock tweets (to be presented in section 2.2) suggests that this view is too simplistic. To start with, one important difference between bullishness/bearishness and positive/negative sentiment is that while the former represents belief about the future, the latter may also refer to the past or present. For example, a user reporting on making profit from a buying stock yesterday and selling it today is clearly positive about the stock, but does not express any prediction about its future performance. Furthermore, messages that do refer to the future differ considerably in their significance. A tweet reporting on buying a stock by the user conveys a much stronger bullishness signal than a tweet that merely expresses an opinion. Overall, it would seem that judging bullishness is far more elusive than judging sentiment.

We therefore propose and compare two alternative approaches that sidestep the complexities of assessing tweets bullishness. These two approaches can be viewed as representing two extremes. The first approach restricts our attention to the most explicit signals of bullishness and bearishness, namely, tweets that report actual buy and sell transactions performed by the user. In the second approach we learn directly the relation between tweets content and stock prices, following previous work on predicting stock price movement from factual sources such as news articles (Lavrenko et al., 2000; Koppel and Shtrimberg, 2004; Schumaker and Chen, 2010). This approach poses no restrictions on the tweets content and avoids any stipulated tweet classification. However, user-generated messages are largely subjective, and their correlation with the stock prices depends on user's expertise. This introduces much noise into the learning process. We show that by making the learning user-sensitive we can improve the results substantially. Overall, our work illustrates the feasibility of finding expert investors, and the utility of following them.

2 Stock Tweets

2.1 Stock Tweets Language

Stock tweets, as Twitter messages in general, are short textual messages of up to 140 characters. They are distinguished by having one or more references to stock symbols (tickers), prefixed by a dollar sign. For instance, the stock of *Apple, Inc.* is referenced as $AAPL. Two other noteworthy Twitter conventions that are also found in stock tweets are *hashtags*, user-defined labels starting with *'#'*, and references to other users, starting with *'@'*. Table 1 lists some examples of stock tweets.

As common with Twitter messages, stock tweets are typically abbreviated and ungrammatical utterances. The language is informal and includes many slang expressions, many of which are unique to the stock tweets community. Thus, many positive and negative expressions common to stock tweets are not found in standard sentiment lexicons. Their unique language and terminology often make stock tweets hard to understand for an outsider. Many words are abbreviated and appear in several non-standard forms. For example, the word *bought* may also appear as *bot* or *bght*, and *today* may appear as *2day*. Stock tweets also contain many sentiment expressions which may appear in many variations, e.g. *wow, woooow, wooooooooow* and so on. These characteristics make the analysis of stock tweets a particularly challenging task.

2.2 Content Analysis

A preliminary step of this research was an extensive data analysis, aimed to gain better understanding of the major types of content conveyed in stock tweets. First, we developed a taxonomy of tweet categories while reading a few thousands of tweets. Based on this taxonomy we then tagged a sample of 350 tweets to obtain statistics on the frequency of each category. The sample contained only tweets that mention exactly one ticker. The following types of tweets were considered irrelevant:

- Tweets that express question. These tweets were labeled as *Question*.
- Obscure tweets, e.g. *"$AAPL fat"*, tweets that contain insufficient information (e.g. *"http://url.com $AAPL"*) and tweets that seem

		Example	%
Fact	News	$KFRC: Deutsche Bank starts at Buy	14.3%
	Chart Pattern	$C (Citigroup Inc) $3.81 crossed its 2nd Pivot Point Support http://empirasign.com/s/x4c	10.9%
	Trade	bot back some $AXP this morning	12.9%
	Trade Outcome	Sold $CELG at 55.80 for day-trade, +0.90 (+1.6%)X	2.9%
Opinion	Speculation	thinking of hedging my shorts by buying some oil. thinking of buying as much $goog as i can in my IRA. but i need more doing, less thinking.	4.0%
	Chart Prediction	http://chart.ly/wsy5ny $GS - not looking good for this one - breaks this support line on volume will nibble a few short	12.9%
	Recommendation	$WFC if you have to own financials, WFC would be my choice. http://fsc.bz/448 #WORDEN	1.7%
	Sentiment	$ivn is rocking	8.6%
Question		$aapl breaking out but in this mkt should wait till close?	7.1%
Irrelevant		$CLNE follow Mr. Clean $$	24.9%

Table 1: Tweets categories and their relative frequencies

to contain no useful information (e.g *"Even Steve Jobs is wrong sometimes... $AAPL http://ow.ly/1Tw0Z"*). These tweets were labeled *Irrelevant*.

The rest of the tweets were classified into two major categories: *Facts* and *Opinions*.

Facts can be divided into four main subcategories:

1. *News:* such tweets are generally in the form of a tweeted headline describing news or a current event generally drawn from mass media. As such they are reliable but, since the information is available in far greater detail elsewhere, their added value is limited.

2. *Chart Pattern:* technical analysis aims to provide insight into trends and emerging patterns in a stock's price. These tweets describe patterns in the stock's chart without the inclusion of any predicted or projected movement, an important contrast to *Chart Prediction*, which is an opinion tweet described below. Chart pattern tweets, like news, are a condensed form of information already available through more in-depth sources and as such their added value is limited.

3. *Trade:* reports an actual purchase or sale of a stock by the user. We consider this as the most valuable form of tweet.

4. *Trade Outcome:* provides details of an "inverse trade", the secondary trade to exit the initial position along with the outcome of the overall trade (profit/loss). The value of these tweets is debatable since although they provide details of a trade, they generally describe the "exit" transaction. This creates a dilemma for analysts since traders will often exit not because of a perceived change in the stock's potential but as a result of many short-term trading activities. For this reason *trade outcome* provides a moderate insight into a user's position which should be viewed with some degree of caution.

Opinions can also be divided into four main subcategories:

1. *Speculation:* provides individual predictions of future events relating to a company or actions of the company. These are amongst the least reliable categories, as the individual user is typically unable to justify his or her insight into the predicted action.

2. *Chart Prediction:* describes a user's prediction of a future chart movement based on technical analysis of the stock's chart.

3. *Recommendation:* As with analyst recommendations, this category represents users who summarize their understanding and insight into

a stock with a simple and effective recommendation to take a certain course of action with regard to a particular share. *Recommendation* is the less determinate counterpart to *Trade*.

4. *Sentiment:* These tweets express pure sentiment toward the stock, rather than any factual content.

Table 1 shows examples for each of the tweet categories, as well as their relative frequency in the analyzed sample.

3 An Expert Finding Framework

In this section we present a general procedure for finding experts in stock microblogs. Based on this procedure, we will develop in the next sections several models for extracting reliable trading signals from tweets.

We assume that a stock tweet refers to exactly one stock, and therefore there is a one-to-one mapping between tweets and stocks. Other tweets are discarded. We define *expertise* as the ability to predict stock rise with high precision. Thus, a user is an *expert* if a high percentage of his or her *bullish* tweets is followed by a stock rise. In principle, we could analogously follow *bearish* tweets, and see if they are followed by a stock fall. However, bearish tweets are somewhat more difficult to interpret: for example, selling a share may indicate a negative outlook on the stock, but it may also result from other considerations, e.g. following a trading strategy that holds the stock for a fixed period (cf. the discussion on *Trade Outcome* tweets in the previous section).

We now describe a procedure that determines whether a user u is an expert. The procedure receives a training set $\mathcal{T}$ of tweets posted by u, where each tweet is annotated with its posting time. It is also given a classifier $\mathcal{C}$, which classifies each tweet as *bullish* or *not bullish* (either bearish or neutral).

The procedure first applies the classifier $\mathcal{C}$ to identify the bullish tweets in $\mathcal{T}$. It then determines the *correctness* of each bullish tweet. Given a tweet t, we observe the price change of the stock referenced by t over a one day period starting at the next trading day. The exact definition of mapping tweets to stock prices is given in section 5.1. A one-day holding period was chosen as it was found to perform well in previous works on tweet-based trading (Zhang and Skiena, 2010; Sprenger and Welpe, 2010), in particular for long positions (buy transactions). A bullish tweet is considered *correct* if it is followed by a stock rise, and as *incorrect* otherwise[1]. Given a set of tweets, we define its *precision* as the percentage of correct tweets in the set. Let C_u, I_u denote the number of correct and incorrect bullish tweets of user u, respectively. The precision of u's bullish tweets is therefore:

$$P_u = \frac{C_u}{C_u + I_u}$$

Let P_{bl} be the baseline precision. In this work we chose the baseline precision to be the proportion of tweets that are followed by a stock rise in the whole training set (including all the users). This represents the expected precision when picking tweets at random. Clearly, if $P_u \leq P_{bl}$ then u is not an expert. If $P_u > P_{bl}$, we apply the following statistical test to assess whether the difference is statistically significant. First, we compute the expected number of correct and incorrect transactions C_{bl}, I_{bl} according to the baseline:

$$C_{bl} = P_{bl} \times (C_u + I_u)$$

$$I_{bl} = (1 - P_{bl}) \times (C_u + I_u)$$

We then compare the observed counts (C_u, I_u) to the expected counts (C_{bl}, I_{bl}), using Pearson's Chi-square test. Since it is required for this test that C_{bl} and I_{bl} are at least 5, cases that do not meet this requirement are discarded. If the resulting p-value satisfies the required significance level α, then u is considered an expert. In this work we take $\alpha = 0.05$. Note that since the statistical test takes into account the number of observations, it will reject cases where the number of the observations is very small, even if the precision is very high. The output of the procedure is a classification of u as expert/non-expert, as well as the p-value (for experts). The expert finding procedure is summarized in Algorithm 1.

In the next two sections we propose several alternatives for the classifier $\mathcal{C}$.

[1] For about 1% of the tweets the stock price did not change in the next trading day. These tweets are also considered *correct* throughout this work.

Algorithm 1 Determine if a user u is an expert

Input: set of tweets $\mathcal{T}$ posted by u, bullishness classifier $\mathcal{C}$, baseline probability P_{bl}, significance level α

Output: NON-EXPERT/(EXPERT, p-value)

$\mathcal{T}_{bullish} \leftarrow$ tweets in $\mathcal{T}$ classified by $\mathcal{C}$ as *bullish*
$C_u \leftarrow 0$; $I_u \leftarrow 0$
for each $t \in \mathcal{T}_{bullish}$ **do**
  if t is followed by a stock rise **then**
    C_u++
  else
    I_u++
  end if
end for
$P_u = \frac{C_u}{C_u + I_u}$
if $P_u \leq P_{bl}$ **then**
  return NON-EXPERT
else
  $C_{bl} \leftarrow P_{bl} \times (C_u + I_u)$
  $I_{bl} \leftarrow (1 - P_{bl}) \times (C_u + I_u)$
  $p \leftarrow$ *ChiSquareTest*(C_u, I_u, C_{bl}, I_{bl})
  if $p > \alpha$ **then**
    return NON-EXPERT
  else
    return (EXPERT, p)
  end if
end if

4 Following Explicit Transactions

The first approach we attempt for classifying bullish (and bearish) tweets aims to identify only tweets that report buy and sell transactions (that is, tweets in the *Trade* category). According to our data analysis (reported in section 2.2), about 13% of the tweets belong to this category. There are two reasons to focus on these tweets. First, as we already noted, actual transactions are clearly the strongest signal of bulishness/bearishness. Second, the buy and sell actions are usually reported using a closed set of expressions, making these tweets relatively easy to identify. A few examples for buy and sell tweets are shown in Table 2.

While buy and sell transactions can be captured reasonably well by a relatively small set of patterns, the examples in Table 2 show that stock tweets have their unique language for reporting these transactions, which must be investigated in order to come by these patterns. Thus, in order to develop a classifier for these tweets, we created a training and test corpora as follows. Based on our preliminary analysis of several thousand tweets, we composed a vocabulary of keywords which trade tweets must include[2]. This vocabulary contained words such as *in, out, bot, bght, sld* and so on. Filtering out tweets that match none of the keywords removed two thirds of the tweets. Out of the remaining tweets, about 5700 tweets were tagged. The training set contains about 3700 tweets, 700 of which are transactions. The test set contains about 2000 tweets, 350 of which are transactions.

sell	sold sum $OMNI 2.14 +12%
buy	bot $MSPD for earnings testing new indicator as well.
sell	Out 1/2 $RIMM calls @ 1.84 (+0.81)
buy	added to $joez 2.56
buy	I picked up some $X JUL 50 Puts @ 3.20 for gap fill play about an hour ago.
buy	long $BIDU 74.01
buy	$$ Anxiously sitting at the bid on $CWCO @ 11.85 It seems the ask and I are at an impasse. 20 min of this so far. Who will budge? (not me)
buy	In 300 $GOOG @ 471.15.
sell	sold $THOR 41.84 for $400 the FreeFactory is rocking
sell	That was quick stopped out $ICE
sell	Initiated a short position in $NEM.

Table 2: Buy and sell tweets

Since the transaction tweets can be characterized by a closed set of recurring patterns, we developed a classifier that is based on a few dozens of manually composed pattern matching rules, formulated as regular expressions. The classifier works in three stages:

1. *Normalization:* The tweet is transformed into a canonical form. For example, user name

[2]That is, we did not come across any trade tweet that does not include at least one of the keywords in the large sample we analyzed, so we assume that such tweets are negligible.

Dataset	Transaction	P	R	F1
Train	Buy	94.0%	84.0%	0.89
	Sell	96.0%	83.0%	0.89
Test	Buy	85.0%	70.0%	0.77
	Sell	88.5%	79.0%	0.84

Table 3: Results for buy/sell transactition classifier. Precision (P), Recall (R), and F-measure (F1) are reported.

is transformed into USERNAME; ticker name is transformed into TICKER; *buy, buying, bought, bot, bght* are transformed into BUY, and so on.

2. *Matching:* Trying to match one of the buy/sell patterns in the normalized tweet.

3. *Filtering:* Filtering out tweets that match "disqualifying" patterns. The simplest examples are a tweet starting with an "if" or a tweet containing a question mark.

The results of the classifier on the train and test set are summarized in Table 3. The results show that our classifier identifies buy/sell transactions with a good precision and a reasonable recall.

5 Unsupervised Learning from Stock Prices

The drawback of the method presented in the previous section is that it only considers a small part of the available tweets. In this section we propose an alternative method, which considers all the available tweets, and does not require any tagged corpus of tweets. Instead, we use actual stock price movements as our labels.

5.1 Associating Tweets with Stock Prices

We used stock prices to label tweets as follows. Each tweet message has a time stamp (eastern time), indicating when it was published. Our policy is to buy in the opening price of the next trading day (P_B), and sell on the opening price of the following trading day (P_S). Tweets that are posted until 9:25 in the morning (market hours begin at 9:30) are associated with the same day, while those are posted after that time are associated with the next trading date.

5.2 Training

Given the buy and sell prices associated with each tweet, we construct positive and negative training examples as follows: positive examples are tweets where $\frac{P_S - P_B}{P_B} \geq 3\%$, and negative examples are tweets where $\frac{P_S - P_B}{P_B} \leq -3\%$.

We used the SVM-light package (Joachims, 1999), with the following features:

- The existence of the following elements in the message text:
  - Reference to a ticker
  - Reference to a user
  - URL
  - Number
  - Hashtag
  - Question mark
- The case-insensitive words in the message after dropping the above elements.
- The 3, 4, 5 letter prefixes of each word.
- The name of the user who authored the tweet, if it is a frequent user (at least 50 messages in the training data). Otherwise, the user name is taken to be "anonymous".
- Whether the stock price was up or down 1% or more in the previous trading day.
- 2, 3, 4-word expressions which are typical to tweets (that is, their relative frequency in tweets is much higher than in general news text).

6 Empirical Evaluation

In this section we focus on the empirical task of *tweet ranking*: ordering the tweets in the test set according to their likelihood to be followed by a stock rise. This is similar to the common IR task of ranking documents according to their relevance. A perfect ranking would place all the correct tweets before all the incorrect ones.

We present several ranking models that use the expert finding framework and the bullishness classification methods discussed in the previous sections as building blocks. The performance of these models is evaluated on the test set. By considering the

precision at various points along the list of ranked tweets, we can compare the precision-recall tradeoffs achieved by each model.

Before we discuss the ranking models and the empirical results, we describe the datasets used to train and test these models.

6.1 Datasets

Stock tweets were downloaded from the StockTwits website[3], during two periods: from April 25, 2010 to November 1, 2011, and from December 14, 2010 to February 3, 2011. A total of 700K tweets messages were downloaded. Tweets that do not contain exactly one stock ticker (traded in NYSE or NASDAQ) were filtered out. The remaining 340K tweets were divided as follows:

- *Development set:* April 25, 2010 to August 31, 2010: 124K messages
- *Held out set*: September 1, 2010 to November 1, 2010: 110K messages
- *Test set:* December 14, 2010 to February 3, 2011: 106K messages

We consider the union of the development and held out sets as our training set.

6.2 Ranking Models

6.2.1 Joint-All Model

This is our baseline model, as it does not attempt to identify experts. It learns a single SVM model as described in Section 5 from all the tweets in the training set. It then applies the SVM model to each tweet in the test set, and ranks them according to the SVM classification score.

6.2.2 Transaction Model

This model finds expert users in the training set (Algorithm 1), using the buy/sell classifier described in Section 4. Tweets classified as *buy* are considered *bullish*, and the rest are considered non-bullish. Expert users are ranked according to their p value (in ascending order). The same classifier is then applied to the tweets of the expert users in the test set. The tweets classified as bullish are ordered according to the ranking of their author (first all the bullish tweets of the highest-ranked expert user, then all the bullish tweets of the expert ranked second, and so on).

6.2.3 Per-User Model

The *joint all* model suffers from the tweets of non-experts twice: at training time, these tweets introduce much noise into the training of the SVM model. At test time, we follow these unreliable tweets along with the more reliable tweets of the experts. The *per-user* model addresses both problems.

This model learns from the development set a separate SVM model $\mathcal{C}_u$ for each user u, based solely on the user's tweets. We then optimize the classification threshold of the learnt SVM model $\mathcal{C}_u$ as follows. Setting the threshold to θ results in a new classifier $\mathcal{C}_{u,\theta}$. Algorithm 1 is applied to u's tweets in the held-out set (denoted $\mathcal{H}_u$), using the classifier $\mathcal{C}_{u,\theta}$. For the ease of presentation, we define *ExpertPValue($\mathcal{H}_u$, $\mathcal{C}_{u,\theta}$,P_{bl},α)* as a function that calls Algorithm 1 with the given parameters, and returns the obtained p-value if u is an expert and 1 otherwise. We search exhaustively for the threshold $\hat{\theta}$ for which this function is minimized (in other words, the threshold that results in the best p-value). The threshold of $\mathcal{C}_u$ is then set to $\hat{\theta}$, and the user's p-value is set to the best p-value found. If u is a non-expert for all of the attempted θ values then u is discarded. Otherwise, u is identified as an expert.

The rest of the process is similar to the transaction model: the tweets of each expert u in the test set are classified using the optimized per-user classifier C_u. The final ranking is obtained by sorting the tweets that were classified as bullish according to the p-value of their author. The per-user ranking procedure is summarized in Algorithm 2.

6.2.4 Joint-Experts Model

The *joint experts* model makes use of the experts identified by the *per-user* model, and builds a single joint SVM model from the tweets of these users. This results in a model that is trained on more examples than in the previous per-user method, but unlike the *joint all* method, it learns only from high-quality users. As with the *joint all* model, test tweets are ranked according to the SVM's score. However, the model considers only the tweets of expert users in the test set.

[3] stocktwits.com

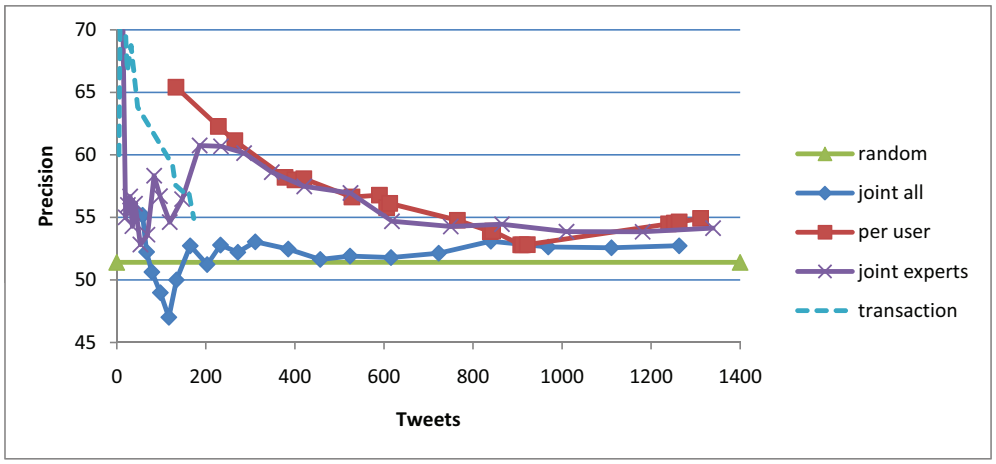


Figure 1: Empirical model comparison

Algorithm 2 Per-user ranking model

Input: dev. set $\mathcal{D}$, held-out set $\mathcal{H}$, test set $\mathcal{S}$, baseline probability P_{bl}, significance level α
Output: A ranked list $\mathcal{R}$ of tweets in $\mathcal{S}$

// Learning from the training set
$E \leftarrow \emptyset$ // set of expert users
for each user u **do**
  $\mathcal{D}_u \leftarrow u$'s tweets in $\mathcal{D}$
  $\mathcal{C}_u \leftarrow$ SVM classifier learnt from $\mathcal{D}_u$
  $\mathcal{H}_u \leftarrow u$'s tweets in $\mathcal{H}$
  $\hat{\theta} = \arg\min_\theta$ *ExpertPValue*($\mathcal{H}_u$, $\mathcal{C}_{u,\theta}$,P_{bl},α)
  $\mathcal{C}_u \leftarrow \mathcal{C}_{u,\hat{\theta}}$
  $p_u \leftarrow$*ExpertPValue*($\mathcal{H}_u$, $\mathcal{C}_{u,\hat{\theta}}$,$P_{bl}$,$\alpha$)
  if $p_u \leq \alpha$ **then**
    add u to E
  end if
end for

// Classifying and ranking the test set
for each user $u \in E$ **do**
  $\mathcal{S}_{bullish,u} \leftarrow u$'s tweets in $\mathcal{S}$ that were classified as *bullish* by $\mathcal{C}_u$
end for
$\mathcal{R} \leftarrow$ tweets in $\bigcup_u \mathcal{S}_{bullish,u}$ sorted by p_u
return $\mathcal{R}$

6.3 Results

Figure 1 summarizes the results obtained for the various models. Each model was used to rank the tweets according to the confidence that they predict a positive stock price movement. Each data point corresponds to the precision obtained for the first k tweets ranked by the model, and the results for varying k values illustrate the precision/recall tradeoff of the model. These data points were obtained as follows:

- For methods that learn a single SVM model (*joint all* and *joint experts*), the graph was obtained by decreasing the threshold of the SVM classifier, at fixed intervals of 0.05. For each threshold value, k is the number of tweets classified as bullish by the model.

- For methods that rank the users by their p value and order the tweets accordingly (*transaction* and *per user*), the i-th data point corresponds to the cumulative precision for the tweets classified as bullish by the first i users. For the *per user* method we show the cumulative results for the first 20 users. For the *transaction* method we show all the users that were identified as experts.

The *random* line is our baseline. It shows the expected results for randomly ordering the tweets in the test set. The expected precision at any point is equal to the percentage of tweets in the test set that were followed by a stock rise, which was found to be 51.4%.

We first consider the *joint all* method, which learns a single model from all the tweets. The only

Correct	Incorrect	P	p
87	46	65.4	0.001
142	86	62.3	0.001
162	103	61.1	0.002
220	158	58.2	0.008
232	168	58.0	0.008
244	176	58.1	0.006
299	229	56.6	0.016
335	255	56.8	0.009
338	268	55.8	0.031
344	269	56.1	0.019
419	346	54.8	0.062
452	387	53.9	0.152
455	389	53.9	0.145
479	428	52.8	0.395
481	430	52.8	0.398
487	435	52.8	0.388
675	564	54.5	0.030
683	569	54.6	0.026
690	573	54.6	0.022
720	591	54.9	0.011

Table 4: Per user model: cumulative results for first 20 users. The table lists the number of correct and incorrect tweets, the precision P and the significance level p.

per-user information available to this model is a feature fed to the SVM classifier, which, as we found, does not contribute to the results. Except for the first 58 tweets, which achieved precision of 55%, the precision quickly dropped to a level of around 52%, which is just a little better than the random baseline. Next, we consider the *transaction* configuration, which is based on detecting *buy* transactions. Only 10 users were found to be experts according to this method, and in the test period these users had a total of 173 tweets. These 173 tweets achieve good precision (57.1% for the first 161 tweets, and 54.9% for the first 173 tweets). However this method resulted in a low number of transactions. This happens because it is able to utilize only a small fraction of the tweets (explicit buy transactions).

Remarkably, *per user* and *joint experts*, the two methods which rely on identifying the experts via unsupervised learning are by far the best methods. Both models seem to have comparable performance, where the results of the *join experts* model are somewhat smoother, as expected. Table 4 shows cumulative results for the first 20 users in the per-user model. The results show that this model achieves good precision for a relatively large number of tweets, and for most of the data points reported in the table the results significantly outperform the baseline (as indicated by the p value). Overall, these results show the effectiveness of our methods for finding experts through unsupervised learning.

7 Related Work

A growing body of work aims at extracting sentiment and opinions from tweets, and exploit this information in a variety of application domains. Davidov et al. (2010) propose utilizing twitter hashtag and smileys to learn enhanced sentiment types. O'Connor et al. (2010) propose a sentiment detector based on Twitter data that may be used as a replacement for public opinion polls. Bollen et al. (2011) measure six different dimensions of public mood from a very large tweet collection, and show that some of these dimensions improve the predication of changes in the Dow Jones Industrial Average (DJIA).

Sentiment analysis of news articles and financial blogs and their application for stock prediction were the subject of several studies in recent years. Some of these works focus on document-level sentiment classification (Devitt and Ahmad, 2007; O'Hare et al., 2009). Other works also aimed at predicting stock movement (Lavrenko et al., 2000; Koppel and Shtrimberg, 2004; Schumaker and Chen, 2010). All these methods rely on predefined sentiment lexicons, manually classified training texts, or their combination. Lavrenko et al. (2000), Koppel and Shtrimberg (2004), and Schumaker and Chen (2010) exploit stock prices for training, and thus save the need in supervised learning.

Previous work on stock message boards include (Das and Chen, 2007; Antweiler and Frank, 2004; Chua et al., 2009). (Sprenger and Welpe, 2010) is, to the best of our knowledge, the first work to address specifically stock microblogs. All these works take a similar approach for classifying message bullishness: they train a classifier (Naïve Bayes, which Das and Chen combined with additional classifiers and a sentiment lexicon, and Chua et al. presented improvement for) on a collection of manually labeled messages (classified into *Buy, Sell, Hold*). Interestingly, Chua et al. made use of an Australian mes-

sage board (HotCopper), where, unlike most of the stock message boards, these labels are added by the message author. Another related work is (Zhang and Skiena, 2010), who apply lexicon-based sentiment analysis to several sources of news and blogs, including tweets. However, their data set does not include stock microblogs, but tweets mentioning the official company name.

Our work differs from previous work on stock messages in two vital aspects. Firstly, these works did not attempt to distinguish between experts and non-expert users, but aggregated the sentiment over all the users when studying the relation between sentiment and the stock market. Secondly, unlike these works, our best-performing methods are completely unsupervised, and require no manually tagged training data or sentiment lexicons.

8 Conclusion

This paper investigated the novel task of finding expert investors in online stock forums. In particular, we focused on stock microblogs. We proposed a framework for finding expert investors, and experimented with several methods for tweet classification using this framework. We found that combining our framework with user-specific unsupervised learning allows us to predict stock price movement with high precision, and the results were shown to be statistically significant. Our results illustrate the importance of distinguishing experts from non-experts. An additional contribution of this work is an in-depth analysis of stock tweets, which sheds light on their content and its potential utility.

In future work we plan to improve the features of the SVM classifier, and further investigate the usefulness of our approach for trading.

References

Werner Antweiler and Murray Z. Frank. 2004. Is all that talk just noise? the information content of internet stock message boards. *Journal of Finance*, 59(3):1259–1294.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1 – 8.

Christopher Chua, Maria Milosavljevic, and James R. Curran. 2009. A sentiment detection engine for internet stock message boards. In *Proceedings of the Australasian Language Technology Association Workshop 2009*.

Sanjiv R. Das and Mike Y. Chen. 2007. Yahoo! for Amazon: Sentiment extraction from small talk on the Web. *Management Science*, 53(9):1375–1388.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10. Association for Computational Linguistics.

Ann Devitt and Khurshid Ahmad. 2007. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 984–991.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

Moshe Koppel and Itai Shtrimberg. 2004. Good news or bad news? Let the market decide. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. 2000. Mining of concurrent text and time series. In *Proceedings of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.

Neil O'Hare, Michael Davy, Adam Bermingham, Paul Ferguson, Pvraic Sheridan, Cathal Gurrin, and Alan F Smeaton. 2009. Topic-dependent sentiment analysis of financial blogs. In *TSA'09 - 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion Measurement*.

Robert P. Schumaker and Hsinchun Chen. 2010. A discrete stock price prediction engine based on financial news. *Computer*, 43:51–56.

Timm O. Sprenger and Isabell M. Welpe. 2010. Tweets and trades: The information content of stock microblogs. Technical report, TUM School of Management, December. working paper.

Wenbin Zhang and Steven Skiena. 2010. Trading strategies to exploit blog and news sentiment. In *ICWSM'10*.

Unsupervised Semantic Role Induction with Graph Partitioning

Joel Lang and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
J.Lang-3@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we present a method for *unsupervised* semantic role induction which we formalize as a graph partitioning problem. Argument instances of a verb are represented as vertices in a graph whose edge weights quantify their role-semantic similarity. Graph partitioning is realized with an algorithm that iteratively assigns vertices to clusters based on the cluster assignments of neighboring vertices. Our method is algorithmically and conceptually simple, especially with respect to how problem-specific knowledge is incorporated into the model. Experimental results on the CoNLL 2008 benchmark dataset demonstrate that our model is competitive with other unsupervised approaches in terms of F1 whilst attaining significantly higher cluster purity.

1 Introduction

Recent years have seen increased interest in the *shallow semantic analysis* of natural language text. The term is most commonly used to describe the automatic identification and labeling of the semantic roles conveyed by sentential constituents (Gildea and Jurafsky, 2002). Semantic roles describe the semantic relations that hold between a predicate and its arguments (e.g., "who" did "what" to "whom", "when", "where", and "how") abstracting over surface syntactic configurations.

In the example sentences below, *window* occupies different syntactic positions — it is the object of *broke* in sentences (1a,b), and the subject in (1c) — while bearing the same semantic role, i.e., the physical object affected by the breaking event. Analogously, *ball* is the instrument of *break* both when realized as a prepositional phrase in (1a) and as a subject in (1b).

(1) a. [Jim]$_{A0}$ **broke** the [window]$_{A1}$ with a [ball]$_{A2}$.
b. The [ball]$_{A2}$ **broke** the [window]$_{A1}$.
c. The [window]$_{A1}$ **broke** [last night]$_{TMP}$.

The semantic roles in the examples are labeled in the style of PropBank (Palmer et al., 2005), a broad-coverage human-annotated corpus of semantic roles and their syntactic realizations. Under the PropBank annotation framework (which we will assume throughout this paper) each predicate is associated with a set of core roles (named *A*0, *A*1, *A*2, and so on) whose interpretations are specific to that predicate[1] and a set of adjunct roles such as *location* or *time* whose interpretation is common across predicates (e.g., *last night* in sentence (1c)).

The availability of PropBank and related resources (e.g., FrameNet; Ruppenhofer et al. (2006)) has sparked the development of great many semantic role labeling systems most of which conceptualize the task as a supervised learning problem and rely on role-annotated data for model training. Most of these systems implement a two-stage architecture consisting of *argument identification* (determining the arguments of the verbal predicate) and *argument classification* (labeling these arguments with semantic roles). Despite being relatively shallow, se-

[1] More precisely, *A*0 and *A*1 have a common interpretation across predicates as *proto-agent* and *proto-patient* in the sense of Dowty (1991).

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1320–1331,
Edinburgh, Scotland, UK, July 27–31, 2011.

mantic role analysis has the potential of benefiting a wide spectrum of applications ranging from information extraction (Surdeanu et al., 2003) and question answering (Shen and Lapata, 2007), to machine translation (Wu and Fung, 2009) and summarization (Melli et al., 2005).

Current approaches have high performance — a system will recall around 81% of the arguments correctly and 95% of those will be assigned a correct semantic role (see Màrquez et al. (2008) for details), however only on languages and domains for which large amounts of role-annotated training data are available. For instance, systems trained on PropBank demonstrate a marked decrease in performance (approximately by 10%) when tested on out-of-domain data (Pradhan et al., 2008). Unfortunately, the reliance on role-annotated data which is expensive and time-consuming to produce for every language and domain, presents a major bottleneck to the widespread application of semantic role labeling.

In this paper we argue that unsupervised methods offer a promising yet challenging alternative. If successful, such methods could lead to significant savings in terms of annotation effort and ultimately yield more portable semantic role labelers that require overall less engineering effort. Our approach formalizes semantic role induction as a graph partitioning problem. Given a verbal predicate, it constructs a weighted graph whose vertices correspond to argument instances of the verb and whose edge weights quantify the similarity between these instances. The graph is partitioned into vertex clusters representing semantic roles using a variant of Chinese Whispers, a graph-clustering algorithm proposed by Biemann (2006). The algorithm iteratively assigns cluster labels to graph vertices by greedily choosing the most common label amongst the neighbors of the vertex being updated. Beyond extending Chinese Whispers to the semantic role induction task, we also show how it can be understood as a type of Gibbs sampling when our graph is interpreted as a Markov random field.

Experimental results on the CoNLL 2008 benchmark dataset demonstrate that our method, despite its simplicity, improves upon competitive approaches in terms of F1 and achieves significantly higher cluster purity.

2 Related Work

Although the bulk of previous work on semantic role labeling has primarily focused on supervised methods (Màrquez et al., 2008), a few semi-supervised and unsupervised approaches have been proposed in the literature. The majority of semi-supervised models have been developed within a framework known as *annotation projection*. The idea is to combine labeled and unlabeled data by projecting annotations from a labeled source sentence onto an unlabeled target sentence within the same language (Fürstenau and Lapata, 2009) or across different languages (Padó and Lapata, 2009). Outwith annotation projection, Gordon and Swanson (2007) propose to increase the coverage of PropBank to unseen verbs by finding syntactically similar (labeled) verbs and using their annotations as surrogate training data.

Swier and Stevenson (2004) were the first to introduce an unsupervised semantic role labeling system. Their algorithm induces role labels following a bootstrapping scheme where the set of labeled instances is iteratively expanded using a classifier trained on previously labeled instances. Their method starts with a dataset containing no role annotations at all, but crucially relies on VerbNet (Kipper et al., 2000) for identifying the arguments of predicates and making initial role assignments. VerbNet is a manually constructed lexicon of verb classes each of which is explicitly associated with argument realization and semantic role specifications.

Subsequent work has focused on unsupervised methods for argument identification and classification. Abend et al. (2009) recognize the arguments of predicates by relying solely on part of speech annotations whereas Abend and Rappoport (2010) distinguish between core and adjunct roles, using an unsupervised parser and part-of-speech tagger. Grenager and Manning (2006) address the role induction problem and propose a directed graphical model which relates a verb, its semantic roles, and their possible syntactic realizations. Latent variables represent the semantic roles of arguments and role induction corresponds to inferring the state of these latent variables.

Following up on this work, Lang and Lapata (2010) formulate role induction as the process of de-

tecting alternations and finding a canonical syntactic form for them. Verbal arguments are then assigned roles, according to their position in this canonical form, since each position references a specific role. Their model extends the logistic classifier with hidden variables and is trained in a manner that takes advantage of the close relationship between syntactic functions and semantic roles. More recently, Lang and Lapata (2011) propose a clustering algorithm which first splits the argument instances of a verb into fine-grained clusters based on syntactic cues and then executes a series of merge steps (mainly) based on lexical cues. The split phase creates a large number of small clusters with high purity but low collocation, i.e., while the instances in a particular cluster typically belong to the same role the instances for a particular role are commonly scattered amongst many clusters. The subsequent merge phase conflates clusters with the same role in order to increase collocation.

Like Grenager and Manning (2006) and Lang and Lapata (2010; 2011), this paper describes an unsupervised method for semantic role induction, i.e., one that does not require any role annotated data or additional semantic resources for training. Contrary to these previous approaches, we conceptualize role induction in a novel way, as a graph partitioning problem. Our method is simple, computationally efficient, and does not rely on hidden variables. Moreover, the graph-based representation for verbs and their arguments affords greater modeling flexibility. A wide range of methods exist for finding partitions in graphs (Schaeffer, 2007), besides Chinese Whispers (Biemann, 2006), which could be easily applied to the semantic role induction problem. However, we leave this to future work.

Graph-based methods are popular in natural language processing, especially with unsupervised learning problems (Chen and Ji, 2010). The Chinese Whispers algorithm itself (Biemann, 2006) has been previously applied to several tasks including word sense induction (Klapaftis and M., 2010) and unsupervised part-of-speech tagging (Christodoulopoulos et al., 2010). The same algorithm is also described in Abney (2007, pp. 146-147) under the name "clustering by propagation". The term makes explicit the algorithm's connection to label propagation, a general framework[2] for semi-supervised learning (Zhu et al., 2003) with applications to machine translation (Alexandrescu and Kirchhoff, 2009), information extraction (Talukdar and Pereira, 2010) and structured part-of-speech tagging (Subramanya et al., 2010). The basic idea behind label propagation is to represent labeled and unlabeled instances as vertices in an undirected graph with edges whose weights express similarity (and possibly dissimilarity) between the instances. Label information is then propagated between the vertices in such a way that similar instances tend to be assigned the same label. Analogously, Chinese Whispers works by propagating cluster membership information along the edges of a graph, even though the graph does not contain any human-labeled instance vertices.

3 Problem Setting

We adopt the standard architecture of supervised semantic role labeling systems where argument identification and argument classification are treated separately. Our role labeler is fully unsupervised with respect to both tasks — it does not rely on any role annotated data or semantic resources. However, our system does not learn from raw text. In common with most semantic role labeling research, we assume that the input is syntactically analyzed in the form of dependency trees.

We view argument identification as a syntactic processing step that can be largely undertaken deterministically through structural analysis of the dependency tree. We therefore use a small set of rules to detect arguments with high precision and recall (see Section 4). Argument classification is more challenging and must take into account syntactic *as well as* lexical-semantic information. Both types of information are incorporated into our model through a similarity function that assigns similarity scores to pairs of argument instances. Following previous work (Lang and Lapata, 2010; Grenager and Manning, 2006), our system outputs verb-specific roles by grouping argument instances into clusters and labeling each argument instance with an identifier cor-

[2]For example, Haffari and Sarkar (2007) use label propagation to analyze other semi-supervised algorithms such as the Yarowsky (1995) algorithm.

responding to the cluster it has been assigned to. Such identifiers are similar to PropBank-style core labels (e.g., *A*0, *A*1).

4 Argument Identification

Supervised semantic role labelers often employ a classifier in order to decide for each node in the parse tree whether or not it represents a semantic argument. Nodes classified as arguments are then assigned a semantic role. In the unsupervised setting, we slightly reformulate argument identification as the task of discarding as many non-semantic arguments as possible. This means that the argument identification component does not make a final positive decision for any of the argument candidates; instead, a final decision is only made in the subsequent argument classification stage.

We discard or select argument candidates using the set of rules developed in Lang and Lapata (2011). These are mainly based on the parts of speech and syntactic relations encountered when traversing a dependency tree from the predicate node to the argument node. For each candidate, rules are considered in a prespecified order and the first matching rule is applied. When evaluated on its own, the argument identification component obtained 88.1% precision (percentage of semantic arguments out of those identified) and 87.9% recall (percentage of identified arguments out of all gold arguments).

5 Argument Classification

After identifying likely arguments for each verb, the next step is to infer a label for each argument instance. Since we aim to induce verb-specific roles (see Section 3), we construct an undirected, weighted graph *for each verb*. Vertices correspond to verb argument instances and edge weights quantify the similarities between them. This argument-instance graph is then partitioned into clusters of vertices representing semantic roles and each argument instance is assigned a label that indicates the cluster it belongs to. In what follows we first describe how the graph is constructed and then provide the details of our graph partitioning algorithm.

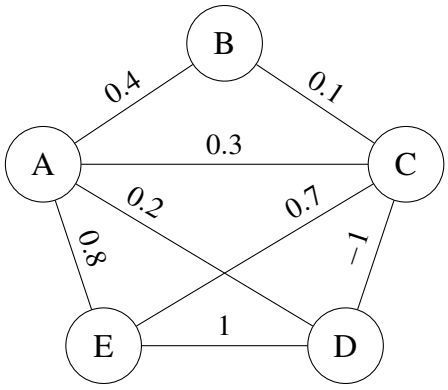


Figure 1: Simplified example of an argument-instance graph. All pairs of vertices with non-zero similarity are connected through edges that are weighted with a similarity score $\phi(v_i, v_j)$. Upon updating the label for a vertex all neighboring vertices propagate their label to the vertex being updated. The score for each label is determined by summing together the weighted votes for that label and the label with the maximal score is chosen.

5.1 Graph Construction

For each verb we construct an undirected, weighted graph $G = (V, E, \phi)$ with vertices V, edges E, and edge weight function ϕ as follows. Each argument instance in the corpus that belongs to the verb is added as a vertex. Then, for each possible pair of vertices (v_i, v_j) we compute a weight $\phi(v_i, v_j) \in \mathbb{R}$ according to the function ϕ. If the weight is non-zero, an undirected edge $e = (v_i, v_j)$ with weight $\phi(v_i, v_j)$ is added to the graph. The function ϕ quantifies the similarity or dissimilarity between instances; positive values indicate that roles are likely to be the same, negative values indicate that roles are likely to differ, and zero values indicate that there is no evidence for either case. Our similarity function is symmetric, i.e., $\phi(v_i, v_j) = \phi(v_j, v_i)$ and permits negative values (see Section 5.4 for a detailed description).

Figure 1 shows an example of a graph for a verb with five argument instances (vertices A–E). Edges are drawn between pairs of vertices with non-zero similarity values. For instance, vertex D is connected to vertex A with weight 0.2, to vertex E with 1, and vertex C with -1. Since edges are drawn between all pairs of vertices with non-zero similarity, the resulting graphs tend to be densely connected, which for large datasets may be prohibitively

inefficient. A solution would be to sample a subset from all possible pairs, but we did not make use of any kind of edge pruning in our experiments.

5.2 Graph Partitioning

Graph partitioning is realized with a variant of Chinese Whispers (Biemann, 2006) whose details are given below. In addition, we discuss how our algorithm relates to other graph-based models in order to help provide a better theoretical understanding.

We assume each vertex v_i is assigned a label $l_i \in \{1 \dots L\}$ indicating the cluster it belongs to. Initially, each vertex belongs to its own cluster, i.e., we let the number of clusters $L = |V|$ and set $l_i \leftarrow i$. Given this initial vertex labeling, the algorithm proceeds by iteratively updating the label for each vertex. The update is based on the labels of neighboring vertices and reflects their similarity to the vertex being updated. Intuitively, each neighboring vertex votes for the cluster it is currently assigned to, where the strength of the vote is determined by the similarity (i.e., edge weight) to the vertex being updated. The label l_i of vertex v_i is thus updated according to the following equation:

$$l_i \leftarrow \arg \max_{l \in \{1 \dots L\}} \sum_{v_j \in \mathcal{N}_i(l)} \phi(v_i, v_j) \tag{2}$$

where $\mathcal{N}_i(l) = \{v_j | (v_i, v_j) \in E \wedge l = l_j\}$ denotes the set of v_i's neighbors with label l. In other words, for each label we compute a score by summing together the weights of edges to neighboring vertices with that label and select the label with the maximal score. Note that negative edges decrease the score for a particular label, thus demoting the label.

Consider again Figure 1. Assume we wish to update vertex A. In addition, assume that B and E are currently assigned the same label (i.e., they belong to the same cluster) whereas C and D are each in different clusters. The score for cluster $\{B,E\}$ is $0.4 + 0.8 = 1.2$, the score for cluster $\{C\}$ is 0.3 and the score for cluster $\{D\}$ is 0.2. We would thus assign A to cluster $\{B,E\}$ as it has the highest score.

The algorithm is run for several iterations. At each iteration it passes over all vertices, and the update order of the vertices is chosen randomly. As the updates proceed, labels can disappear from the graph, whereby the number of clusters decreases. Empirically, we observe that for sufficiently many iterations the algorithm converges to a fixed labeling or oscillates between labelings that differ only in a few vertices. The result of the algorithm is a hard partitioning of the given graph, where the number of clusters is determined automatically.

5.3 Propagation Prioritization

We make one important modification to the basic algorithm described so far based on the intuition that higher scores for a label indicate more reliable propagations. More precisely, when updating vertex v_i to label l we define the confidence of the update as the average similarity to neighbors with label l:

$$conf(l_i \leftarrow l) = \frac{1}{|\mathcal{N}_i(l)|} \sum_{v_j \in \mathcal{N}_i(l)} \phi(v_i, v_j) \tag{3}$$

We can then prioritize high-confidence updates by setting a threshold θ and allowing only updates with confidence greater or equal to θ. The threshold is initially set to 1 (i.e., the maximal possible confidence) and then lowered by some small constant Δ after each iteration until it reaches a minimum θ_{min}, at which point the algorithm terminates. This improves the resulting clustering, since it promotes reliable updates in earlier phases of the algorithm which in turn has a positive effect on successive updates.

5.4 Argument-Instance Similarity

As described earlier, the edge weights in our graph are similarity scores, with positive values indicating similarity and negative values indicating dissimilarity. Determining the similarity function ϕ without access to labeled training data poses a major difficulty which we resolve by relying on prior linguistic knowledge. Specifically, we measure the similarity of argument instances based on three simple and intuitive criteria: (1) whether the instances are lexically similar; (2) whether the instances occur in the same syntactic position; and (3) whether the instances occur in the same frame (i.e., are arguments in the same clause). The same criteria were used in (Lang and Lapata, 2011) and shown effective in quantifying role-semantic similarity between *clusters* of argument instances. Lexical and syntactic similarity are scored through functions $lex(v_i, v_j)$ and $syn(v_i, v_j)$ with range $[-1, 1]$, whereas the third criterion enters the scoring function directly:

$$\phi(v_i, v_j) = \begin{cases} -\infty \text{ if } v_i \text{ and } v_j \text{ are in same frame} \\ \alpha lex(v_i, v_j) + (1-\alpha) syn(v_i, v_j) \text{ otherwise.} \end{cases} \quad (4)$$

The first case in the function expresses a common linguistic assumption, i.e., that two argument instances v_i and v_j occurring in the same frame cannot have the same semantic role. The function implements this constraint by returning $-\infty$.[3] The syntactic similarity function $s(v_i, v_j)$ indicates whether two argument instances occur in a similar syntactic position. We define syntactic positions through four cues: the relation of the argument head word to its governor, verb voice (active/passive), the linear position of the argument relative to the verb (left/right) and the preposition used for realizing the argument (if any). The score is $\frac{S}{4}$ where S is the number of cues which agree, i.e., have the same value. The syntactic score is set to zero when the governor relation of the arguments is not the same. Lexical similarity $l(v_i, v_j)$ is measured in terms of the cosine of the angle between vectors h_i and h_j representing the argument head words:

$$lex(v_i, v_j) = cos(h_i, h_j) = \frac{h_i \cdot h_j}{\|h_i\| \|h_j\|} \quad (5)$$

We obtain h_i and h_j from a simple semantic space model (Turney and Pantel, 2010) which requires no supervision (Section 6 describes the details of the model used in our experiments).

Our similarity function weights the contribution of syntax vs. semantics equally, i.e., α is set to 0.5. This reflects the linguistic intuition that lexical and syntactic information are roughly of equal importance.

5.5 Relation to Other Models

This section briefly points out some connections to related models. The averaging procedure used for updating the graph vertices (Equation 2) appears in some form in most label propagation algorithms (see Talukdar (2010) for details). Label propagation algorithms are commonly interpreted as random walks on graphs. In our case such an interpretation is not directly possible due to the presence of negative edge weights. This could be changed by transforming the edge weights onto a non-negative scale, but we find the current setup more expedient for modeling dissimilarity.

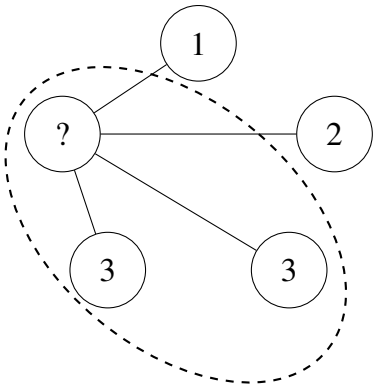


Figure 2: The update rule (Equation 2) can be understood as choosing a minimal edge-cut, thereby greedily maximizing intra-cluster similarity and minimizing inter-cluster similarity. Assuming equal weight for all edges above, label 3 is chosen for the vertex being updated such that the sum of weights of edges crossing the cut is minimal.

Our model could be also transformed into a probabilistic graphical model that specifies a distribution over vertex labels. In the transformed model each vertex corresponds to a random variable over labels and edges are associated with binary potential functions over vertex-pairs. Let $\mathbf{1}(v_i = v_j)$ denote an indicator function which takes value 1 iff. $l_i = l_j$ and value 0, otherwise. Then pairwise potentials can be defined in terms of the original edge weights[4] as $\psi(v_i, v_j) = \exp(\mathbf{1}(v_i = v_j)\phi(v_i, v_j))$. A Gibbs sampler used to sample from the distribution of the resulting pairwise Markov random field (Bishop, 2006; Wainwright and Jordan, 2008) would employ almost the same update procedure as in Equation 2, the difference being that labels would be sampled according to their probabilities, rather than chosen deterministically based on scores.

A third way of understanding the update rule is as a heuristic for maximizing intra-cluster similarity and minimizing inter-cluster similarity. By

[3] Formally, ϕ has range $ran(\phi) = [-1, 1] \cup \{-\infty\}$ and for $x \in ran(\phi)$ we define $x + (-\infty) = -\infty$. This means that the overall score computed for a label (Equation 2) is $-\infty$ if one of the summands is $-\infty$.

[4] Including weights with value zero and thus connecting all vertex pairs.

assigning the label with maximal score to v_i, we greedily maximize the sum of intra-cluster edge weights while minimizing the sum of inter-cluster edge weights, i.e., the weight of the edge-cut. This is illustrated in Figure 2. Cut-based methods are a common method in graph clustering (Schaeffer, 2007) and are also used for inference in pairwise Markov random fields like the one described in the previous paragraph (Boykov et al., 2001).

Note that while it would be possible to transform our model into a model with a formal probabilistic interpretation (either as a graph random walk or as a probabilistic graphical model) this would not change the non-empirical nature of the similarity function, which is unavoidable in the unsupervised setting and is also common in the semi-supervised methods discussed in Section 2.

6 Experimental Setup

In this section we describe how we assessed the performance of our model. We discuss the dataset on which our experiments were carried out, explain how our system's output was evaluated and present the methods used for comparison with our approach.

Data We compared the output of our model against the PropBank gold standard annotations contained in the CoNLL 2008 shared task dataset (Surdeanu et al., 2008). The latter was taken from the Wall Street Journal portion of the Penn Treebank and converted into a dependency format (Surdeanu et al., 2008). In addition to gold standard dependency parses, the dataset also contains automatic parses obtained from the MaltParser (Nivre et al., 2007). The dataset provides annotations for verbal and nominal predicate-argument constructions, but we only considered the former, following previous work on semantic role labeling (Màrquez et al., 2008). All the experiments described in this paper use the CoNLL 2008 training dataset.

Evaluation Metrics For each verb, we determine the extent to which argument instances in the clusters share the same gold standard role (purity) and the extent to which a particular gold standard role is assigned to a single cluster (collocation).

More formally, for each group of verb-specific clusters we measure the purity of the clusters as the percentage of instances belonging to the majority gold class in their respective cluster. Let N denote the total number of instances, G_j the set of instances belonging to the j-th gold class and C_i the set of instances belonging to the i-th cluster. Purity can be then written as:

$$PU = \frac{1}{N} \sum_i \max_j |G_j \cap C_i| \tag{6}$$

Collocation is defined as follows. For each gold role, we determine the cluster with the largest number of instances for that role (the role's *primary* cluster) and then compute the percentage of instances that belong to the primary cluster for each gold role:

$$CO = \frac{1}{N} \sum_j \max_i |G_j \cap C_i| \tag{7}$$

Per-verb scores are aggregated into an overall score by averaging over all verbs. We use the micro-average obtained by weighting the scores for individual verbs proportionately to the number of instances for that verb.

Finally, we use the harmonic mean of purity and collocation as a single measure of clustering quality:

$$F_1 = \frac{2 \cdot CO \cdot PU}{CO + PU} \tag{8}$$

Model Parameters Recall that our algorithm prioritizes updates with confidence higher than a threshold θ. Initially, θ is set to 1 and its value decreases at each iteration by a small constant Δ which we set to 0.0025. The algorithm terminates when a minimum confidence θ_{min} is reached. While choosing a value for Δ is straightforward — it simply has to be a small fraction of the maximal possible confidence — specifying θ_{min} on the basis of objective prior knowledge is less so. And although a human judge could determine the optimal termination point based on several criteria such as clustering quality or the number of clusters, we used a development set instead for the sake of reproducibility and comparability. Specifically, we optimized θ_{min} on the CoNLL test set and obtained best results with $\theta_{min} = \frac{1}{3}$. This value was used for all our experiments and was also kept fixed for all verbs. Importantly, the development set was not used for any kind of supervised training.

	Syntactic Function			Latent Logistic			Split-Merge			Graph Partitioning		
	PU	CO	F1	PU	CO	F1	PU	CO	F1	PU	CO	F1
auto/auto	72.9	73.9	73.4	73.2	**76.0**	74.6	81.9	71.2	**76.2**	**82.5**	68.8	75.0
gold/auto	77.7	**80.1**	**78.9**	75.6	79.4	77.4	**84.0**	74.4	**78.9**	**84.0**	73.5	78.4
auto/gold	77.0	71.0	73.9	77.9	**74.4**	76.2	86.5	69.8	**77.3**	**87.4**	65.9	75.2
gold/gold	81.6	**77.5**	79.5	79.5	76.5	78.0	**88.7**	73.0	**80.1**	88.6	70.7	78.6

Table 1: Evaluation of the output of our graph partitioning algorithm compared to our previous models and a baseline that assigns arguments to clusters based on their syntactic function.

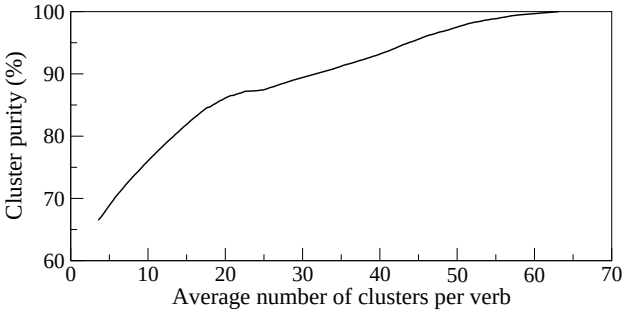


Figure 3: Purity (vertical axis) against average number of clusters per verb (horizontal axis) on the auto/auto dataset.

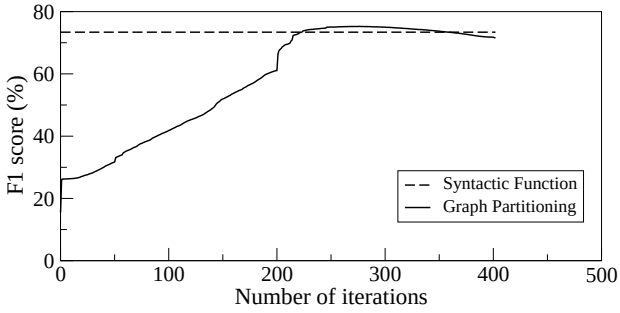


Figure 4: F1 (vertical axis) against number of iterations (horizontal axis) on the auto/auto dataset.

Recall that one of the components in our similarity function is lexical similarity which we measure using a vector-based model (see Section 5.4). We created such a model from the Google *N*-Grams corpus (Brants and Franz, 2006) using a context window of two words on both sides of the target word and co-occurrence frequencies as vector components (no weighting was applied). The large size of this corpus allows us to use bigram frequencies, rather than frequencies of individual words and to distinguish between left and right bigrams. We used randomized algorithms (Ravichandran et al., 2005) to build the semantic space efficiently.

Comparison Models We compared our graph partitioning algorithm against three competitive approaches. The first one assigns argument instances to clusters according to their syntactic function (e.g., subject, object) as determined by a parser. This baseline has been previously used as a point of comparison by other unsupervised semantic role induction systems (Grenager and Manning, 2006; Lang and Lapata, 2010) and shown difficult to outperform. Our implementation allocates up to $N = 21$ clusters[5] for each verb, one for each of the 20 most frequent syntactic functions and a default cluster for all other functions. We also compared our approach to Lang and Lapata (2010) using the same model settings (with 10 latent variables) and feature set proposed in that paper. Finally, our third comparison model is Lang and Lapata's (2011) split-merge clustering algorithm. Again we used the same parameters and number of clusters (on average 10 per verb). Our graph partitioning method uses identical cues for assessing role-semantic similarity as the method described in Lang and Lapata (2011).

7 Results

Our results are summarized in Table 1. We report cluster purity (PU), collocation (CO) and their harmonic mean (F1) for the baseline (Syntactic Function), our two previous models (the Latent Logistic classifier and Split-Merge) and the graph partitioning algorithm on four datasets. These result from the combination of automatic parses with automatically identified arguments (auto/auto), gold parses with

[5] This is the number of gold standard roles.

Syntactic Function												
PU	91.4	68.6	45.1	59.7	62.4	61.9	63.5	75.9	76.7	69.6	63.1	53.7
CO	91.3	71.9	56.0	68.4	72.7	76.8	65.6	79.7	76.0	63.8	73.4	58.9
F1	91.4	70.2	49.9	63.7	67.1	68.6	64.5	77.7	76.3	66.6	67.9	56.2
Graph Partitioning												
PU	95.6	83.5	72.3	75.4	83.3	84.4	74.8	84.8	89.5	83.0	73.2	66.3
CO	89.1	62.7	42.1	64.2	56.2	66.3	57.2	73.2	64.1	54.3	66.0	57.7
F1	92.2	71.6	53.2	69.4	67.1	74.3	64.8	78.5	74.7	65.7	69.4	61.7
Verb	say	make	go	increase	know	tell	consider	acquire	meet	send	open	break
Freq	15238	4250	2109	1392	983	911	753	704	574	506	482	246

Table 2: Clustering results for individual verbs on the auto/auto dataset with our graph partitioning algorithm and the syntactic function baseline; the scores were taken from a single run.

automatic arguments (gold/auto), automatic parses with gold arguments (auto/gold) and gold parses with gold arguments (gold/gold). Table 1 reports averages across multiple runs. This was necessary in order to ensure that the results of our randomized graph partitioning algorithm are stable.[6] The arguments for the auto/auto and gold/auto datasets were identified using the rules described in Lang and Lapata (2011) (see Section 4). Bold-face is used to highlight the best performing system under each measure (PU, CO, or F1) on each dataset.

Compared to the Syntactic Function baseline, the Graph Partitioning algorithm has higher F1 on the auto/auto and auto/gold datasets but lags behind by 0.5 points on the gold/auto dataset and by 0.9 points on the gold/gold dataset. It attains highest purity on all datasets except for gold/gold, where it is 0.1 points below Split-Merge. When considering F1 in conjunction with purity and collocation, we observe that Graph Partitioning can attain higher purity than the comparison models by trading off collocation. If we were to hand label the clusters output by our system, purity would correspond to the quality of the resulting labeling, while collocation would determine the labeling effort. The relationship is illustrated more explicitly in Figure 3, which plots purity against the average number of clusters per verb on the auto/auto dataset. As the algorithm proceeds the number of clusters is reduced which results in a decrease of purity. The latter decreases more rapidly once the number of 20 clusters per verb is reached. This is accompanied by a decreasing tradeoff ratio between collocation and purity: at this stage decreasing purity by one point increases collocation by roughly one point, whereas in earlier iterations a decrease of purity by one point goes together with several points increase in collocation. This is most likely due to the fact that the number of gold standard classes is around 20.

Figure 4 shows the complete learning curve of our graph partitioning method on the auto/auto dataset (F1 is plotted against the number of iterations). The algorithm naturally terminates at iteration 266 (when $\theta_{min} = 1/3$), but we have also plotted iterations beyond that point. Since lower values of θ permit unreliable propagations, F1 eventually falls below the baseline (see Section 5.2). The importance of our propagation prioritization mechanism is further underlined by the fact that when it is not employed (i.e., when using the vanilla Chinese Whispers algorithm without any modifications), it performs substantially worse than the comparison models. On the auto/auto dataset, F1 converges to 59.1 (purity is 55.5 and collocation 63.2) within 10 iterations.

Finally, Table 2 shows how performance varies across verbs. We report results for the Syntactic Function baseline and Graph Partitioning on the auto/auto dataset for 12 verbs. These were selected so as to exhibit varied occurrence frequencies and alternation patterns. As can be seen, the macro-

[6] For example, on the auto/auto dataset and over 10 runs, the standard deviation in F1 was 0.11 points in collocation 0.16 points and in purity 0.08 points. The worst F1 was 0.20 points below the average, the worst collocation was 0.32 points below the average and the worst purity was 0.17 points below the average.

scopic result — increase in F1 and purity — also holds across verbs.

8 Conclusions

In this paper we described an unsupervised method for semantic role induction, in which argument-instance graphs are partitioned into clusters representing semantic roles. The approach is conceptually and algorithmically simple and novel in its formalization of role induction as a graph partitioning problem. We believe this constitutes an interesting alternative for two reasons. Firstly, eliciting and encoding problem-specific knowledge in the form of instance-wise similarity judgments can be easier than encoding it into model structure e.g., by making statistical independence assumptions or assumptions about latent structure. Secondly, the approach is general and amenable to other graph partitioning algorithms and relates to well-known graph-based semi-supervised learning methods.

The similarity function in this paper is by necessity rudimentary, since it cannot be estimated from data. Nevertheless, the resulting system attains competitive F1 and notably higher purity than the comparison models. Arguably, performance could be improved by developing a better similarity function. Therefore, in the future we intend to investigate how our system performs in a weakly supervised setting, where the similarity function is estimated from a small amount of labeled instances, since this would allow us to incorporate richer syntactic features and result in more precise similarity scores.

Acknowledgments We are grateful to Charles Sutton for his valuable feedback on this work. The authors acknowledge the support of EPSRC (grant GR/T04540/01).

References

O. Abend and A. Rappoport. 2010. Fully unsupervised core-adjunct argument classification. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 226–236, Uppsala, Sweden.

O. Abend, R. Reichart, and A. Rappoport. 2009. Unsupervised Argument Identification for Semantic Role Labeling. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 28–36, Singapore.

S. Abney. 2007. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC.

A. Alexandrescu and K. Kirchhoff. 2009. Graph-based learning for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 119–127, Boulder, Colorado.

C. Biemann. 2006. Chinese Whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York City.

C. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

T. Brants and A. Franz. 2006. Web 1T 5-gram Version 1. Linguistic Data Consortium, Philadelphia.

Z. Chen and H. Ji. 2010. Graph-based clustering for computational linguistics: A survey. In *Proceedings of TextGraphs-5 - 2010 Workshop on Graph-based Methods for Natural Language Processing*, pages 1–9, Uppsala, Sweden.

C. Christodoulopoulos, S. Goldwater, and M. Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA.

D. Dowty. 1991. Thematic Proto Roles and Argument Selection. *Language*, 67(3):547–619.

H. Fürstenau and M. Lapata. 2009. Graph Aligment for Semi-Supervised Semantic Role Labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 11–20, Singapore.

D. Gildea and D. Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

A. Gordon and R. Swanson. 2007. Generalizing Semantic Role Annotations Across Syntactically Similar Verbs. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 192–199, Prague, Czech Republic.

T. Grenager and C. Manning. 2006. Unsupervised Discovery of a Statistical Verb Lexicon. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 1–8, Sydney, Australia.

G. Haffari and A. Sarkar. 2007. Analysis of Semi-Supervised Learning with the Yarowsky Algorithm. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, Vancouver, BC.

K. Kipper, H. T. Dang, and M. Palmer. 2000. Class-Based Construction of a Verb Lexicon. In *Proceedings of the 17th AAAI Conference on Artificial Intelligence*, pages 691–696. AAAI Press / The MIT Press.

I. Klapaftis and Suresh M. 2010. Word sense induction & disambiguation using hierarchical random graphs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 745–755, Cambridge, MA.

J. Lang and M. Lapata. 2010. Unsupervised Induction of Semantic Roles. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, California.

J. Lang and M. Lapata. 2011. Unsupervised Semantic Role Induction via Split-Merge Clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon. To appear in.

L. Màrquez, X. Carreras, K. Litkowski, and S. Stevenson. 2008. Semantic Role Labeling: an Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159.

G. Melli, Y. Wang, Y. Liu, M. M. Kashani, Z. Shi, B. Gu, A. Sarkar, and F. Popowich. 2005. Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2005 Summarization Task. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing Document Understanding Workshop*, Vancouver, Canada.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit A. Chanev, S. Kübler, S. Marinov, and E. Marsi. 2007. MaltParser: A Language-independent System for Data-driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135.

S. Padó and M. Lapata. 2009. Cross-lingual Annotation Projection of Semantic Roles. *Journal of Artificial Intelligence Research*, 36:307–340.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

S. Pradhan, W. Ward, and J. Martin. 2008. Towards Robust Semantic Role Labeling. *Computational Linguistics*, 34(2):289–310.

D. Ravichandran, P. Pantel, and E. Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Function for High Speed Noun Clustering. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, page 622629, Ann Arbor, Michigan.

J. Ruppenhofer, M. Ellsworth, M. Petruck, C. Johnson, and J. Scheffczyk. 2006. FrameNet II: Extended Theory and Practice, version 1.3. Technical report, International Computer Science Institute, Berkeley, CA, USA.

S. Schaeffer. 2007. Graph clustering. *Computer Science Review*, 1(1):27–64.

D. Shen and M. Lapata. 2007. Using Semantic Roles to Improve Question Answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning*, pages 12–21, Prague, Czech Republic.

A. Subramanya, S. Petrov, and F. Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176, Cambridge, MA.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo, Japan.

M. Surdeanu, R. Johansson, A. Meyers, and L. Màrquez. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th CoNLL*, pages 159–177, Manchester, England.

R. Swier and S. Stevenson. 2004. Unsupervised Semantic Role Labelling. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 95–102, Barcelona, Spain.

P. Talukdar and F. Pereira. 2010. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1473–1481, Uppsala, Sweden.

P. Talukdar. 2010. *Graph-Based Weakly Supervised Methods for Information Extraction & Integration*. Ph.D. thesis, CIS Department, University of Pennsylvania.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

M. Wainwright and M. Jordan. 2008. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

D. Wu and P. Fung. 2009. Semantic Roles for SMT: A Hybrid Two-Pass Model. In *Proceedings of North*

American Annual Meeting of the Association for Computational Linguistics HLT 2009: Short Papers, pages 13–16, Boulder, Colorado.

D. Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the 20th International Conference on Machine Learning*, Washington, DC.

Structural Opinion Mining for Graph-based Sentiment Representation

Yuanbin Wu, Qi Zhang, Xuanjing Huang, Lide Wu
Fudan University
School of Computer Science
{ybwu,qz,xjhuang,ldwu}@fudan.edu.cn

Abstract

Based on analysis of on-line review corpus we observe that most sentences have complicated opinion structures and they cannot be well represented by existing methods, such as frame-based and feature-based ones. In this work, we propose a novel graph-based representation for sentence level sentiment. An integer linear programming-based structural learning method is then introduced to produce the graph representations of input sentences. Experimental evaluations on a manually labeled Chinese corpus demonstrate the effectiveness of the proposed approach.

1 Introduction

Sentiment analysis has received much attention in recent years. A number of automatic methods have been proposed to identify and extract opinions, emotions, and sentiments from text. Previous researches on sentiment analysis tackled the problem on various levels of granularity including document, sentence, phrase and word (Pang et al., 2002; Riloff et al., 2003; Dave et al., 2003; Takamura et al., 2005; Kim and Hovy, 2006; Somasundaran et al., 2008; Dasgupta and Ng, 2009; Hassan and Radev, 2010). They mainly focused on two directions: sentiment classification which detects the overall polarity of a text; sentiment related information extraction which tries to answer the questions like "*who* expresses *what* opinion on *which* target".

Most of the current studies on the second direction assume that an opinion can be structured as a frame which is composed of a fixed number of slots. Typical slots include opinion holder, opinion expression, and evaluation target. Under this representation, they defined the task as a slots filling problem for each of the opinions. Named entity recognition and relation extraction techniques are usually applied in this task (Hu and Liu, 2004; Kobayashi et al., 2007; Wu et al., 2009).

However, through data analysis, we observe that 60.5% of sentences in our corpus do not follow the assumption used by them. A lot of important information about an opinion may be lost using those representation methods. Consider the following examples, which are extracted from real online reviews:

Example 1: *The interior is a bit noisy on the freeway*[1].

Example 2: *Takes good pictures during the daytime. Very poor picture quality at night*[2].

Based on the definition of opinion unit proposed by Hu and Liu (2004), from the first example, the information we can get is the author's negative opinion about "interior" using an opinion expression "noisy". However, the important restriction "on the freeway", which narrows the scope of the opinion, is ignored. In fact, the tuple ("noisy","on the freeway") cannot correctly express the original opinion: it is negative but under certain condition. The second example is similar. If the conditions "during the daytime" and "at night" are dropped, the extracted elements cannot correctly represent user's opinions.

Example 3: *The camera is actually quite good for outdoors because of the software*.

Besides that, an opinion expression may induce other opinions which are not expressed directly. In example 3, the opinion expression is "good" whose

[1] http://reviews.carreview.com/blog/2010-ford-focus-review-the-compact-car-that-can/

[2] http://www.dooyoo.co.uk/digital-camera/sony-cyber-shot-dsc-s500/1151680/

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1332–1341,
Edinburgh, Scotland, UK, July 27–31, 2011.

target is "camera". But the "software" which triggers the opinion expression "good" is also endowed with a positive opinion. In practice, this induced opinion on "software" is actually more informative than its direct counterpart. Mining those opinions may help to form a complete sentiment analysis result.

Example 4: *The image quality is in the middle of its class, but it can still be a reasonable choice for students.*

Furthermore, the relations among individual opinions also provide additional information which is lost when they are considered separately. Example 4 is such a case that the whole positive comment of camera is expressed by a transition from a negative opinion to a positive one.

In order to address those issues, this paper describes a novel sentiment representation and analysis method. Our main contributions are as follows:

1. We investigate the use of graphs for representing sentence level sentiment. The vertices are evaluation target, opinion expression, modifiers of opinion. The Edges represent relations among them. The semantic relations among individual opinions are also included. Through the graph, various information on opinion expressions which is ignored by current representation methods can be well handled. And the proposed representation is language-independent.

2. We propose a supervised structural learning method which takes a sentence as input and the proposed sentiment representation for it as output. The inference algorithm is based on integer linear programming which helps to concisely and uniformly handle various properties of our sentiment representation. By setting appropriate prior substructure constraints of the graph, the whole algorithm achieves reasonable performances.

The remaining part of this paper is organized as follows: In Section 2 we discuss the proposed representation method. Section 3 describes the computational model used to construct it. Experimental results in test collections and analysis are shown in Section 4. In Section 5, we present the related work and Section 6 concludes the paper.

2 Graph-based Sentiment Representation

In this work, we propose using directed graph to represent sentiments. In the graph, vertices are text spans in the sentences which are opinion expressions, evaluation targets, conditional clauses etc. Two types of edges are included in the graph: (1) relations among opinion expressions and their modifiers; (2) relations among opinion expressions. The edges of the first type exist within individual opinions. The second type of the edges captures the relations among individual opinions. The following sections detail the definition.

2.1 Individual Opinion Representation

Let r be an opinion expression in a sentence, the representation unit for r is a set of relations $\{(r, d_k)\}$. For each relation (r, d_k), d_k is a *modifier* which is a span of text specifying the change of r's meaning.

The relations between modifier and opinion expression can be the type of any kind. In this work, we mainly consider two basic types:

- *opinion restriction*. (r, d_k) is called an opinion restriction if d_k narrows r's scope, adds a condition, or places limitations on r's original meaning.

- *opinion expansion*. (r, d_k) is an opinion expansion if r's scope expands to d_k, r induces another opinion on d_k, or the opinion on d_k is implicitly expressed by r.

Mining the opinion restrictions can help to get accurate meaning of an opinion, and the opinion expansions are useful to cover more indirect opinions. As with previous sentiment representations, we actually consider the third type of modifier which d_k is the evaluation target of r.

Figure 1 shows a concrete example. In this example, there are three opinion expressions: "good", "sharp", "slightly soft". The modifiers of "good" are "indoors" and "Focus accuracy", where relation ("good","indoors") is an opinion restriction because "indoors" is the condition under which "Focus accuracy" is good. On the other hand, the relation

("sharp", "little 3x optical zooms") is an opinion expansion because the "sharp" opinion on "shot" implies a positive opinion on "little 3x optical zooms".

It is worth to remark that: 1) a modifier d_k can relate to more than one opinion expression. For example, multiple opinion expressions may share a same condition; 2) d_k itself can employ a set of relations, although the case appears occasionally. The following is an example:

Example 5: *The camera wisely get rid of many redundant buttons.*

In the example, "redundant buttons" is the evaluation target of opinion expression "wisely get rid of", but itself is a relation between "redundant" and "buttons". Such nested semantic structure is described by a path: "wisely get rid of" $\xrightarrow{target}$ ["redundant" $\xrightarrow{target}$ "buttons"]$_{\text{nested target}}$.

2.2 Relations between Individual Opinion Representation

Assume $\langle r_i \rangle$ are opinion expressions ordered by their positions in sentence, and each of them has been represented by relations $\{(r_i, d_{ik})\}$ individually (the nested relations for d_{ik} have also been determined). Then we define two relations on adjacent pair r_i, r_{i+1}: *coordination* when the polarities of r_i and r_{i+1} are consistent, and *transition* when they are opposite. Those relations among r_i form a set B called *opinion thread*. In Figure 1, the opinion thread is: {("good", "sharp"), ("sharp", "slightly soft")}.

The whole sentiment representation for a sentence can be organized by a direct graph $G = (V, E)$. Vertex set V includes all opinion expressions and modifiers. Edge set E collects both relations of each individual opinion and relations in opinion thread. The edges are labeled with relation types in label set $\mathcal{L}$={"restriction", "expansion", "target", "coordination", "transition"} [3].

Compared with previous works, the advantages of using G as sentiment representation are: 1) for individual opinions, the modifiers will collect more information than using opinion expression alone.

[3]We don't define any "label" on vertices: if two span of text satisfy a relation in $\mathcal{L}$, they are chosen to be vertices and an edge with proper label will appear in E. In other words, vertices are identified by checking whether there exist relations among them.

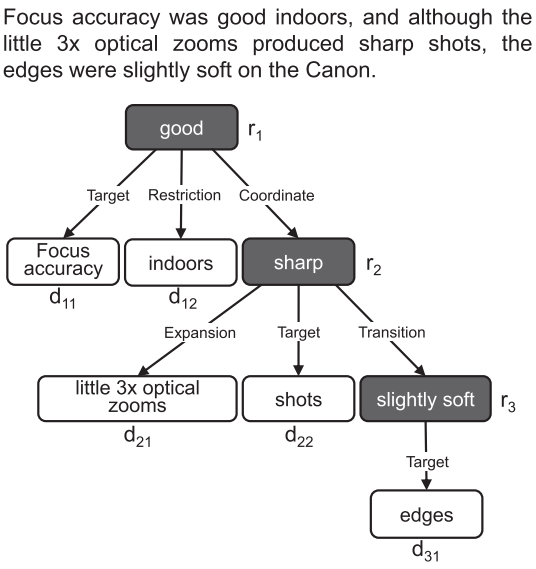


Figure 1: Sentiment representation for an example sentence

Thus G is a relatively complete and accurate representation; 2) the opinion thread can help to catch global sentiment information, for example the general polarity of a sentence, which is dropped when the opinions are separately represented.

3 System Description

To produce the representation graph G for a sentence, we need to extract candidate vertices and build the relations among them to get a graph structure. For the first task, the experimental results in Section 4 demonstrate that the standard sequential labeling method with simple features can achieve reasonable performance. In this section, we focus on the second task, and assume the vertices in the graph have already been correctly collected in the following formulation of algorithm.

3.1 Preliminaries

In order to construct graph G, we use a structural learning method. The framework is from the first order discriminative dependency parsing model (Mcdonald and Pereira, 2005). A sentence is denoted by s; $\mathbf{x}$ are text spans which will be vertices of graph; x_i is the ith vertex in $\mathbf{x}$ ordered by their positions in s. For a set of vertices $\mathbf{x}$, $\mathbf{y}$ is the graph of its sentiment representation, and $e = (x_i, x_j) \in \mathbf{y}$ is the direct edge from x_i to x_j in $\mathbf{y}$. In addition, x_0 is a

virtual root node without inedge. $\mathcal{G} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_n^N$ is training set.

Following the edge based factorization, the score of a graph is the sum of its edges' scores,

$$\begin{aligned} \text{score}(\mathbf{x}, \mathbf{y}) &= \sum_{(x_i,x_j)\in\mathbf{y}} \text{score}(x_i, x_j) \\ &= \sum_{(x_i,x_j)\in\mathbf{y}} \alpha^T \mathbf{f}(x_i, x_j), \end{aligned} \quad (1)$$

$\mathbf{f}(x_i, x_j)$ is a high dimensional feature vector of the edge (x_i, x_j). The components of $\mathbf{f}$ are either 0 or 1. For example the k-th component could be

$$\mathbf{f}_k(x_i, x_j) = \begin{cases} 1 & \text{if } x_i\text{.POS} = \text{JJ and } x_j\text{.POS} = \text{NN} \\ & \quad \text{and label of } (x_i, x_j) \text{is restriction .} \\ 0 & \text{otherwise} \end{cases}$$

Then the score of an edge is the linear combination of $\mathbf{f}$'s components, and the coefficients are in vector α.

Algorithm 1 shows the parameter learning process. It aims to get parameter α which will assign the correct graph $\mathbf{y}$ with the highest score among all possible graphs of $\mathbf{x}$ (denoted by $\mathcal{Y}$).

Algorithm 1 Online structural learning

Training Set:$\mathcal{G} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_n^N$

$\alpha^0 = 0, r = 0, T =$maximum iteration
for $t = 0$ to T **do**
  for $n = 0$ to N **do**
    $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} \text{score}(\mathbf{x}_n, \mathbf{y})$ ▷ Inference
    if $\hat{\mathbf{y}} \neq \mathbf{y}_n$ **then**
      update α^t to α^{t+1} ▷ PA
      $r = r + \alpha^{t+1}$
    end if
  end for
end for
return $\alpha = r/(N * T)$

3.2 Inference

Like other structural learning tasks, the "$\arg\max$" operation in the algorithm (also called inference)

$$\begin{aligned} \hat{\mathbf{y}} &= \arg\max_{\mathbf{y}\in\mathcal{Y}} \text{score}(\mathbf{x}, \mathbf{y}) \\ &= \arg\max_{\mathbf{y}\in\mathcal{Y}} \sum_{(x_i,x_j)\in\mathbf{y}} \alpha^T \mathbf{f}(x_i, x_j) \end{aligned} \quad (2)$$

is hard because all possible values of $\mathbf{y}$ form a huge search space. In our case, $\mathcal{Y}$ is all possible directed acyclic graphs of the given vertex set, which number is exponential. Directly solving the problem of finding maximum weighted acyclic graph is equivalent to finding maximum feedback arc set, which is a NP-hard problem (Karp, 1972). We will use integer linear programming (ILP) as the framework for this inference problem.

3.2.1 Graph Properties

We first show some properties of graph G either from the definition of relations or corpus statistics.

Property 1. The graph is connected and without directed cycle. From individual opinion representation, each subgraph of G which takes an opinion expression as root is connected and acyclic. Thus the connectedness is guaranteed for opinion expressions are connected in opinion thread; the acyclic is guaranteed by the fact that if a modifier is shared by different opinion expressions, the inedges from them always keep (directed) acyclic.

Property 2. Each vertex can have one outedge labeled with coordination or transition at most. The opinion thread B is a directed path in graph.

Property 3. The graph is sparse. The average in-degree of a vertex is 1.03 in our corpus, thus the graph is almost a rooted tree. In other words, the cases that a modifier connects to more than one opinion expression rarely occur comparing with those vertices which have a single parent. An explaination for this sparseness is that opinions in online reviews always concentrate in local context and have local semantic connections.

3.2.2 ILP Formulation

Based on the property 3, we divide the inference algorithm into two steps: i) constructing G's spanning tree (arborescence) with property 1 and 2; ii) finding additional non-tree edges as a post processing task. The first step is close to the works on ILP formulations of dependency parsing (Riedel and Clarke, 2006; Martins et al., 2009). In the second step, we use a heuristic method which greedily adds non-tree edges. A similar approximation method is also used in (Mcdonald and Pereira, 2006) for acyclic dependency graphs.

Step 1. Find MST. Following the multicommodity

flow formulation of maximum spanning tree (MST) problem in (Magnanti and Wolsey, 1994), the ILP for MST is:

$$\text{max.} \sum_{i,j} y_{ij} \cdot \text{score}(x_i, x_j) \quad (3)$$

$$\text{s.t.} \quad \sum_{i,j} y_{ij} = |V| - 1 \quad (4)$$

$$\sum_i f_{ij}^u - \sum_k f_{jk}^u = \delta_j^u, 1 \le u, j \le |V| \quad (5)$$

$$\sum_k f_{0k}^u = 1, \qquad 1 \le u \le |V| \quad (6)$$

$$f_{ij}^u \le y_{ij}, \qquad 1 \le u, j \le |V|, \; 0 \le i \le |V| \quad (7)$$

$$f_{ij}^u \ge 0, \qquad 1 \le u, j \le |V|, \; 0 \le i \le |V| \quad (8)$$

$$y_{ij} \in \{0, 1\}, \qquad 0 \le i, j \le |V|. \quad (9)$$

In this formulation, y_{ij} is an edge indicator variable that (x_i, x_j) is a spanning tree edge when $y_{ij} = 1$, (x_i, x_j) is a non-tree edge when $y_{ij} = 0$. Then output $\mathbf{y}$ is represented by the set $\{y_{ij}, 0 \le i, j \le |V|\}$ [4]. Eq(4) ensures that there will be exactly $|V| - 1$ edges are chosen. Thus if the edges corresponding to those non zero y_{ij} is a connected subgraph, $\mathbf{y}$ is a well-formed spanning tree. Objective function just says the optimal solution of y_{ij} have the maximum weight.

The connectedness is guaranteed if for every vertex, there is exactly one path from root to it. It is formulated by using $|V| - 1$ flows $\{f^u, 1 \le u \le |V|\}$. f^u starts from virtual root x_0 towards vertex x_u. Each flow $f^u = \{f_{ij}^u, 0 \le i, j \le |V|\}$. f_{ij}^u indicates whether flow f^u is through edge (x_i, x_j). so it should be 0 if edge (x_i, x_j) does not exist (by (7)). The Kronecker's delta δ_j^u in (5) guarantees f^u is only assumed by vertex x_u, so f^u is a well-formed path from root to x_u. (6) ensures there is only one flow (path) from root to x_u. Thus the subgraph is connected. The following are our constraints:

c1: Constraint on edges in opinion thread (10)-(11).

From the definition of opinion thread, we impose a constraint on every vertex's outedges in opinion thread, which are labeled with "coordination" or "transition". Let $\mathbb{I}_{ob}$ be a characteristic function on edges: $\mathbb{I}_{ob}((j,k)) = 1$ when edge (x_j, x_k) is labeled with "coordination" or "transition", otherwise 0. We denote q variables for vertices:

$$q_j = \sum_k y_{jk} \cdot \mathbb{I}_{ob}((j,k)), \quad 0 \le j \le |V|. \quad (10)$$

Then following linear inequalities bound the number of outedges in opinion thread (≤ 1) on each vertex:

$$q_j \le 1, \qquad 0 \le j \le |V|. \quad (11)$$

c2: Constraint on target edge (12).

We also bound the number of evaluation targets for a vertex in a similar way. Let $\mathbb{I}_t$ be characteristic function on edges identifing whether it is labeled with "target",

$$\sum_k y_{jk} \cdot \mathbb{I}_t((j,k)) \le C_t, \quad 0 \le j \le |V|. \quad (12)$$

The parameter C_t can be adjusted according to the style of document. In online reviews, authors tend to use simple and short comments on individual targets, so C_t could be set small.

c3: Constraint on opinion thread (13)-(18).

From graph property 2, the opinion thread should be a directed path. It implies the number of connected components whose edges are "coordination" or "transition" should be less than 1. Two set of additional variables are needed: $\{c_j, 0 \le j \le |V|\}$ and $\{h_j, 0 \le j \le |V|\}$, where

$$c_j = \begin{cases} 1 & \text{if an opinion thread starts at } x_j \\ 0 & \text{otherwise} \end{cases},$$

and

$$h_j = \sum_i y_{ij} \cdot \mathbb{I}_{ob}((i,j)). \quad (13)$$

Then $c_j = \neg h_j \wedge q_j$, which can be linearized by

$$c_j \ge q_j - h_j, \quad (14)$$

$$c_j \le 1 - h_j, \quad (15)$$

$$c_j \le q_j, \quad (16)$$

$$c_j \ge 0. \quad (17)$$

If the sum of c_j is no more than 1, the opinion thread of graph is a directed path.

$$\sum_j c_j \le 1. \quad (18)$$

[4] For simplicity, we overload symbol $\mathbf{y}$ from the graph of the sentiment represetation to the MST of it.

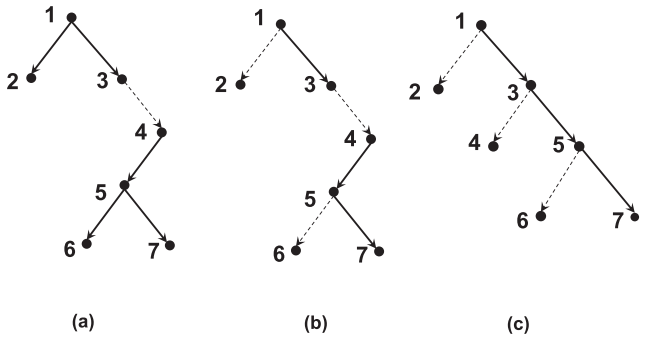


Figure 2: The effects of **c1** and **c3**. Assume solid lines are edges labeled with "coordination" and "transition", dot lines are edges labeled with other types. (a) is an arbitrary tree. (b) is a tree with **c1** constraints. (c) is a tree with **c1** and **c3**. It shows **c1** are not sufficient for graph property 2: the edges in opinion thread may not be connected.

Figure 2 illustrates the effects of **c1** and **c3**.

Equations (10)-(18), together with basic multicommodity flow model build up the inference algorithm. The entire ILP formulation involves $O(|V|^3)$ variables and $O(|V|^2)$ constraints. Generally, ILP falls into NPC, but as an important result, in the multicommodity flow formulation of maximum spanning tree problem, the integer constraints (9) on y_{ij} can be dropped. So the problem reduces to a linear programming which is polynomial solvable (Magnanti and Wolsey, 1994). Unfortunately, with our additional constraints the LP relaxation is not valid.

Step 2. Adding non-tree edges. We examine the case that a modifier attaches to different opinion expressions. That often occurs as the result of the sharing of modifiers among adjacent opinion expressions. We add those edges in the following heuristic way: If a vertex r_i in opinion thread does not have any modifier, we search the modifiers of its adjacent vertices r_{i+1}, r_{i-1} in the opinion thread, and add edge (r_i, d^*) where

$$d^* = \arg\max_{d \in S} \text{score}(r_i, d),$$

and S are the modifiers of r_{i-1} and r_{i+1}.

3.3 Training

We use online passive aggressive algorithm (PA) with Hamming cost of two graphs in training (Crammer et al., 2006).

Unigram Feature Template	
Inside Features	x_i.text w_0.text w_1.text
	w_0.POS w_1.POS
	w_{k-1}.text w_k.text
	w_{k-1}.POS w_k.POS
	x_i.hasDigital
	x_i.isSingleWord
	x_i.hasSentimentWord
	x_i.hasParallelPhrase
Outside Features	w_{-1}.text w_{-2}.text
	w_{-1}.POS w_{-2}.POS
	w_{k+1}.text w_{k+2}.text
	w_{k+1}.POS w_{k+2}.POS
	c_{-1}.text c_{-2}.text
	c_{-1}.POS c_{-2}.POS
	c_{l+1}.text c_{l+2}.text
	c_{l+1}.POS c_{l+2}.POS
Other Features	
distance between parent and child	
dependency parsing relations	

Table 1: Feature set

3.4 Feature Construction

For each vertex x_i in graph, we use 2 sets of features: inside features which are extracted inside the text span of x_i; outside features which are outside the text span of x_i. A vertex x_i is described both in word sequence $(w_0, w_1, \cdots, w_k)$ and character sequence $(c_0, c_1, \cdots, c_l)$, for the sentences are in Chinese.

$$\cdots, w_{-1}, \underbrace{w_0, w_1, w_2, \cdots, w_{k-1}, w_k}_{x_i}, w_{k+1} \cdots$$
$$\cdots, c_{-1}, \underbrace{c_0, c_1, c_2, \cdots, c_{l-1}, c_l}_{x_i}, c_{l+1} \cdots$$

For an edge (x_i, x_j), the high dimensional feature vector $\mathbf{f}(x_i, x_j)$ is generated by using unigram features in Table 1 on x_i and x_j respectively. The distance between parent and child in sentence is also attached in features. In order to involve syntactic information, whether there is certain type of dependency relation between x_i and x_j is also used as a feature.

4 Experiments

4.1 Corpus

We constructed a Chinese online review corpus from Pcpop.com, Zol.com.cn, and It168.com, which have a large number of reviews about digital camera. The corpus contains 138 documents and 1735 sentences. Since some sentences do not contain any opinion, 1390 subjective sentences were finally chosen and manually labeled.

Two annotators labeled the corpus independently. The annotators started from locating opinion expressions, and for each of them, they annotated other modifiers related to it. In order to keep the reliability of annotations, another annotator was asked to check the corpus and determine the conflicts. Finally, we extracted 6103 elements, which are connected by 6284 relations.

Relation	Number
Target	2479
Coordinate	1173
Transition	154
Restriction	693
Expansion	386

Table 2: Statistics of relation types

Table 2 shows the number of various relation types appearing in the labeled corpus. We observe 60.5% of sentences and 32.1% of opinion expressions contain other modifiers besides "target". Thus only mining the relations between opinion expressions and evaluation target is actually at risk of inaccurate and incomplete results.

4.2 Experiments Configurations

In all the experiments below, we take 90% of the corpus as training set, 10% as test set and run 10 folder cross validation. In feature construction, we use an external Chinese sentiment lexicon which contains 4566 positive opinion words and 4370 negative opinion words. For Chinese word segment, we use ctbparser [5]. Stanford parser (Klein and Manning, 2003) is used for dependency parsing. In the settings of PA, the maximum iteration number is set to 2, which is chosen by maximizing the testing performances, aggressiveness parameter C is set to 0.00001. For parameters in inference algorithm, $C_t = 2$, the solver of ILP is lpsolve[6].

We evaluate the system from the following aspects: 1) whether the structural information helps to mining opinion relations. 2) How the proposed inference algorithm performs with different constraints. 3) How the various features affect the system. Except for the last one, the feature set used for different experiments are the same ("In+Out+Dep" in Table 5). The criteria for evaluation are similar to the unlabeled attachment score in parser evaluations, but due to the equation $|E| = |V| - 1$ is not valid if G is not a tree, we evaluate precision $P = \frac{\texttt{\#true edges in result graph}}{\texttt{\#edges in result graph}}$, recall $R = \frac{\texttt{\#true edges in result graph}}{\texttt{\#edges in true graph}}$, and F-score $F = \frac{2P \cdot R}{P+R}$.

4.3 Results

1. *The effects of structural information.* An alternative method to extract relations is directly using a classifier to judge whether there is a relation between any two elements. Those kinds of methods were used in previous opinion mining works (Wu et al., 2009; Kobayashi et al., 2007). To show the entire structural information is important for mining relations, we use SVM for binary classification on candidate pairs. The data point representing a pair (x_i, x_j) is the same as the high dimensional feature vectors $\mathbf{f}(x_i, x_j)$. The setting of our algorithm "MST+c1+c2+c3" is the basic MST with all the constraints. The results are shown in the Table 3.

	P	R	F
SVM	64.9	24.0	35.0
MST+c1+c2+c3-m	61.5	74.0	67.2
MST+c1+c2+c3	73.1	71.0	72.1

Table 3: Binary classifier and structural learning

From the results, the performance of SVM (especially recall) is relatively poor. A possible reason is that the huge imbalance of positive and negative training samples (only $\Theta(n)$ positive pairs among all n^2 pairs). And the absence of global structural

[5] http://code.google.com/p/ctbparser/

[6] http://sourceforge.net/projects/lpsolve/

knowledge makes binary classifier unable to use the information provided by classification results of other pairs.

In order to examine whether the complicated sentiment representation would disturb the classifier in finding relations between opinion expressions and its target, we evaluate the system by discarding the modifiers of opinion restriction and expansion from the corpus. The result is shown in the second row of Table 3. We observe that "MST+c1+c2+c3" is still better which means at least on overall performance the additional modifiers do not harm.

2. *The effect of constraints on inference algorithm*. In the inference algorithm, we utilized the properties of graph G and adapted the basic multicommodity flow ILP to our specific task. To evaluate how the constraints affect the system, we decompose the algorithm and combine them in different ways.

	P	R	F
MST	69.3	67.3	68.3
MST+c1	70.0	68.0	69.0
MST+c2	69.8	67.8	68.8
MST+c1+c2	70.6	68.6	69.6
MST+c1+c3	72.4	70.4	71.4
MST+c1+c2+c3	73.1	71.0	72.1
MST+c1+c2+c3+g	72.5	72.3	72.4

Table 4: Results on inference methods. "MST" is the basic multicommodity flow formulation of maximum spanning tree; c1, c2, c3 are groups of constraint from Section 3.2.2; "g" is our heuristic method for additional non spanning tree edges.

From Table 4, we observe that with any additional constraints the inference algorithm outperforms the basic maximum spanning tree method. It implies although we did not use high order model (e.g. involving grandparent and sibling features), prior structural constraints can also help to get a better output graph. By comparing with different constraint combinations, the constraints on opinion thread (c1, c3) are more effective than constraints on evaluation targets (c2). It is because opinion expressions are more important in the entire sentiment representation. The main structure of a graph is clear once the relations between opinion expressions are correctly determined.

3. *The effects of various features*. We evaluate the performances of different feature configurations in Table 5. From the results, the outside feature set is more effective than inside feature set, even if it does not use any external resource. A possible reason is that the content of a vertex can be very complicated (a vertex even can be a clause), but the features surrounding the vertex are relatively simple and easy to identify (for example, a single preposition can identify a complex condition). The dependency feature has limited effect, due to that lots of online review sentences are ungrammatical and parsing results are unreliable. And the complexity of vertices also messes the dependency feature.

	P	R	F
In-s	66.3	66.3	66.3
In	66.7	66.4	66.6
Out	67.8	67.4	67.6
In+Out	72.0	70.5	71.0
In+Out+Dep	72.5	72.3	72.4

Table 5: Results with different features. "In" represents the result of inside feature set; "In-s" is "In" without the external opinion lexicon feature; "Out" uses the outside feature set; "In+Out" uses both "In" and "Out", "In+Out+Dep" adds the dependency feature. The inference algorithm is "MST+c1+c2+c3+g" in Table 4.

We analyze the errors in test results. A main source of errors is the confusion of classifier between "target" relations and "coordination", "transition" relations. The reason may be that for a modification on opinion expression (r, d_k), we allow d_k recursively has its own modifiers (Example 5). Thus an opinion expression can be a modifier which brings difficulties to classifier.

4. *Extraction of vertices.* Finally we conduct an experiment on vertex extraction using standard sequential labeling method. The tag set is simply {B, I, O} which are signs of begin, inside, outside of a vertex. The underlying model is conditional random field [7]. Feature templates involved are in Table 6. We only use basic features in the experiment. 10 folder cross validation results are in table 7. We suspect that the performances (especially recall) could be improved if some external resources(i.e. ontology, domain related lexicon, etc.) are involved.

[7]We use CRF++ toolkit, http://crfpp.sourceforge.net/

Unigram Template	
c_i.char	character
c_i.isDigit	digit
c_i.isAlpha	english letter
c_i.isPunc	punctuation
c_i.inDict	in a sentiment word
c_i.BWord	start of a word
c_i.EWord	end of a word

Table 6: Features for vertex extraction. The sequential labeling is conducted on character level (c_i). The sentiment lexicon used in c_i.inDict is the same as Table1. We also use bigram feature templates on c_i.char, c_i.isAlpha, c_i.inDict with respect to c_{i-1} and c_{i+1}.

	P	R	F
E+Unigram	56.8	45.1	50.3
E+Unigram+Bigram	57.3	47.9	52.1
O+Unigram	71.9	57.2	63.7
O+Unigram+Bigram	72.3	60.2	65.6

Table 7: Results on vertices extraction with 10 folder cross validation. We use two criterion: 1) the vertex is correct if it is exactly same as ground truth("E"), 2) the vertex is correct if it overlaps with ground truth("O").

5 Related Work

Opinion mining has recently received considerable attentions. Large amount of work has been done on sentimental classification in different levels and sentiment related information extraction. Researches on different types of sentences such as comparative sentences (Jindal and Liu, 2006) and conditional sentences (Narayanan et al., 2009) have also been proposed.

Kobayashi et al. (2007) presented their work on extracting opinion units including: opinion holder, subject, aspect and evaluation. They used slots to represent evaluations, converted the task to two kinds of relation extraction tasks and proposed a machine learning-based method which used both contextual and statistical clues.

Jindal and Liu (2006) studied the problem of identifying comparative sentences. They analyzed different types of comparative sentences and proposed learning approaches to identify them.

Sentiment analysis of conditional sentences were studied by Narayanan et al. (2009). They aimed to determine whether opinions expressed on different topics in a conditional sentence are positive, negative or neutral. They analyzed the conditional sentences in both linguistic and computational perspectives and used learning method to do it. They followed the *feature-based sentiment analysis* model (Hu and Liu, 2004), which also use flat frames to represent evaluations.

Integer linear programming was used in many NLP tasks (Denis and Baldridge, 2007), for its power in both expressing and approximating various inference problems, especially in parsing (Riedel and Clarke, 2006; Martins et al., 2009). Martins etc. (2009) also applied ILP with flow formulation for maximum spanning tree, besides, they also handled dependency parse trees involving high order features(sibling, grandparent), and with projective constraint.

6 Conclusions

This paper introduces a representation method for opinions in online reviews. Inspections on corpus show that the information ignored in previous sentiment representation can cause incorrect or incomplete mining results. We consider opinion restriction, opinion expansions, relations between opinion expressions, and represent them with a directed graph. Structural learning method is used to produce the graph for a sentence. An inference algorithm is proposed based on the properties of the graph. Experimental evaluations with a manually labeled corpus are given to show the importance of structural information and effectiveness of proposed inference algorithm.

7 Acknowledgement

The author wishes to thank the anonymous reviewers for their helpful comments. This work was partially funded by 973 Program (2010CB327906), National Natural Science Foundation of China (61003092, 61073069),863 Program of China (2009AA01A346), Shanghai Science and Technology Development Funds(10dz1500104), Doctoral Fund of Ministry of Education of China (200802460066), Shanghai Leading Academic Discipline Project (B114), and Key Projects in the National Science & Technology Pillar Pro-

gram(2009BAH40B04).

References

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Sajib Dasgupta and Vincent Ng. 2009. Mine the easy, classify the hard: A semi-supervised approach to automatic sentiment classification. In *Proceedings of ACL-IJCNLP*.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of WWW*.

Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of NAACL-HLT*.

Ahmed Hassan and Dragomir R. Radev. 2010. Identifying text polarity using random walks. In *Proceedings of ACL*, pages 395–403, Uppsala, Sweden, July. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of SIGKDD*.

Nitin Jindal and Bing Liu. 2006. Identifying comparative sentences in text documents. In *Proceedings of SIGIR*.

R. Karp. 1972. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING-ACL*.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS*, pages 3–10. MIT Press.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *Proceedings of EMNLP-CoNLL*.

Thomas L. Magnanti and Laurence A. Wolsey. 1994. Optimal trees.

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL-IJCNLP*.

R. Mcdonald and F. Pereira. 2005. Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics*.

Ryan Mcdonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*, pages 81–88.

Ramanathan Narayanan, Bing Liu, and Alok Choudhary. 2009. Sentiment analysis of conditional sentences. In *Proceedings of EMNLP*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proc. of EMNLP 2002*.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of EMNLP*.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL*.

Swapna Somasundaran, Janyce Wiebe, and Josef Ruppenhofer. 2008. Discourse level opinion interpretation. In *Proceedings of COLING*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of ACL*.

Yuanbin Wu, Qi Zhang, Xuangjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of EMNLP*.

Summarize What You Are Interested In: An Optimization Framework for Interactive Personalized Summarization

Rui Yan
Department of Computer Science and Technology, Peking University, Beijing 100871, China
r.yan@pku.edu.cn

Jian-Yun Nie
Département d´informatique et de recherche opérationnelle, Université de Montréal, Montréal, H3C 3J7 Québec, Canada
nie@iro.umontreal.ca

Xiaoming Li
Department of Computer Science and Technology, Peking University, Beijing 100871, China
lxm@pku.edu.cn

Abstract

Most traditional summarization methods treat their outputs as static and plain texts, which fail to capture user interests during summarization because the generated summaries are the same for different users. However, users have individual preferences on a particular source document collection and obviously a universal summary for all users might not always be satisfactory. Hence we investigate an important and challenging problem in summary generation, i.e., Interactive Personalized Summarization (IPS), which generates summaries in an interactive and personalized manner. Given the source documents, IPS captures user interests by enabling interactive clicks and incorporates personalization by modeling captured reader preference. We develop experimental systems to compare 5 rival algorithms on 4 instinctively different datasets which amount to 5197 documents. Evaluation results in ROUGE metrics indicate the comparable performance between IPS and the best competing system but IPS produces summaries with much more user satisfaction according to evaluator ratings. Besides, low ROUGE consistency among these user preferred summaries indicates the existence of personalization.

1 Introduction

In the era of information explosion, people need new information to update their knowledge whilst information on Web is updating extremely fast. Multi-document summarization has been proposed to address such dilemma by producing a summary delivering the majority of information content from a document set, and hence is a necessity.

Traditional summarization methods play an important role with the exponential document growth on the Web. However, for the readers, the impact of human interests has seldom been considered. Traditional summarization utilizes the same methodology to generate the same summary no matter who is reading. However, users may have bias on what they prefer to read due to their potential interests: they need *personalization*. Therefore, traditional summarization methods are to some extent insufficient.

Topic biased summarization tries for personalization by pre-defining human interests as several general categories, such as *health* or *science*. Readers are required to select their possible interests before summary generation so that the chosen topic has priority during summarization. Unfortunately, such topic biased summarization is not sufficient for two reasons: (1) interests cannot usually be accurately pre-defined by ambiguous topic categories and (2) user interests cannot always be foreknown. Often users do not really know what general ideas or detail information they are interested in until they read the summaries. Therefore, more flexible *interactions* are required to establish personalization.

Due to all the insufficiencies of existed summarization approaches, we introduce a new multi-document summarization task of Interactive Personalized Summarization (IPS) and a novel solution for the task. Taking a document collection as input, the system outputs a summary aligned both with source corpus and with user personalization, which is captured by flexible *human−system* interactions. We

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1342–1351,
Edinburgh, Scotland, UK, July 27–31, 2011.

build an experimental system on 4 real datasets to verify the effectiveness of our methods compared with 4 rivals. The contribution of IPS is manifold by addressing following challenges:

• The **1st challenge** for IPS is to integrate user interests into traditional summary components. We measure the utilities of these components and combine them. We formulate the task into a balanced optimization framework via iterative substitution to generate summaries with maximum overall utilities.

• The **2nd challenge** is to capture user interests through interaction. We develop an interactive mechanism of "click" and "examine" between readers and summaries and address sparse data by "click smoothing" under the scenario of few user clicks.

We start by reviewing previous works. In Section 3 we provide IPS overview, describe user interaction and optimize component combination with personalization. We conduct empirical evaluation and demonstrate the experimental system in Section 4. Finally we draw conclusions in Section 5.

2 Related Work

Multi-Document Summarization (MDS) has drawn much attention in recent years and gained emphasis in conferences such as ACL, EMNLP and SIGIR, etc. General MDS can either be extractive or abstractive. The former assigns salient scores to semantic units (e.g. sentences, paragraphs) of the documents indicating their importance and then extracts top ranked ones, while the latter demands information fusion(e.g. sentence compression and reformulation). Here we focus on extractive summarization.

Centroid-based method is one of the most popular extractive summarization method. MEAD (Radev et al., 2004) and NeATS (Lin and Hovy, 2002) are such implementations, using position and term frequency, etc. MMR (Goldstein et al., 1999) algorithm is used to remove redundancy. Most recently, the graph-based ranking methods have been proposed to rank sentences or passages based on the "votes" or "recommendations" between each other. The graph-based methods first construct a graph representing the sentence relationships at different granularities and then evaluate the saliency score of the sentences based on the graph. TextRank (Mihalcea and Tarau, 2005) and LexPageRank (Erkan and Radev, 2004) use algorithms similar to PageRank and HITS to compute sentence importance. Wan et al. improve the graph-ranking algorithm by differentiating intra-document and inter-document links between sentences (2007b) and incorporate cluster information in the graph model to evaluate sentences (2008).

To date, topics (or themes, clusters) in documents have been discovered and used for sentence selection for topic biased summarization (Wan and Yang, 2008; Gong and Liu, 2001). Wan et al. have proposed a manifold-ranking method to make uniform use of sentence-to-sentence and sentence-to-topic relationships to generate topic biased summaries (2007a). Leuski et al. in (2003) pre-define several topic concepts, assuming users will foresee their interested topics and then generate the topic biased summary. However, such assumption is not quite reasonable because user interests may not be forecasted, or pre-defined accurately as we have explained in last section.

The above algorithms are usually traditional extensions of generic summarizers. They do not involve interactive mechanisms to capture reader interests, nor do they utilize user preference for personalization in summarization. Wan et al. in (2008) have proposed a summarization biased to neighboring reading context through anchor texts. However, such scenario does not apply to contexts without human-edited anchor texts like Wikipedia they have used. Our approach can naturally and simultaneously take into account traditional summary elements and user interests and combine both in optimization under a wider practical scenario.

3 Interactive Personalized Summarization

Personalization based on user preference can be captured via various alternative ways, such as *eye-tracking* or *mouse-tracking* instruments used in (Guo and Agichtein, 2010). In this study, we utilize interactive user clicks/examinations for personalization.

Unlike traditional summarization, IPS supports *human−system* interaction by *clicking* into the summary sentences and *examining* source contexts. The implicit feedback of user clicks indicates what they are interested in and the system collects preference information to update summaries if readers wish to. We obtain an associated tuple $<q, c>$ between a

clicked sentence **q** and the examined contexts **c**.

As q has close semantic coherence with neighboring contexts due to consistency in human natural language, we consider a window of sentences centered at the clicked sentence q as c, which is a bag of sentences. The window size k is a parameter to set.

However, click data is often sparse: users are not likely to click more than 1/10 of total summary sentences within a single generation. We amplify these tiny hints of user interest by *click smoothing*.

We change the flat summary structure into a hierarchical organization by extracting important semantic units (denoted as **u**) and establishing linkage between them. If the clicked sentence q contains u, we diffuse the click impact to the correlated units, which makes a single click perform as multiple clicks and the sparse data is smoothed.

Problem Formulation

Input: Given the sentence collection D decomposed by documents, $D = \{s_1, s_2, \dots, s_{|D|}\}$ and the clicked sentence record $Q = \{q_1, q_2, \dots\}$, we generate summaries in sentences. A user click is associated with a tuple $<q, (u), c>$ where the existence of u depends on whether q contains u. The collection of semantic units is denoted as $M = \{u_1, u_2, \dots, u_{|M|}\}$.

Output: A summary S as a set of sentences $\{s_1, s_2, \dots, s_{|S|}\}$ and $S \subset D$ according to the prespecified compression rate ϕ ($0 < \phi < 1$).

After the overview and formulation of IPS problem, we move on to the major components of *User Interaction* and *Personalized Summarization*.

3.1 User Interaction

Hypertexify Summaries. We hypertexify the summary structure by establishing linkage between semantic units. There are several possible formats for semantic units, such as words or n-grams, etc. As single words are proved to be not illustrative of semantic meanings (Zhao et al., 2011) and n-grams are rigid in length, we choose to extract semantic units at a phrase granularity. Among all phrases from source texts, some are of higher importance to attract user interests, such as hot concepts or popular event names. We utilize the toolkit provided by (Zhao et al., 2011) based on graph proximity LDA (Blei et al., 2003) to extract key phrases and their corresponding topic. A topic T is represented by $\{(u_1, \pi(u_1, T)), (u_2, \pi(u_2, T)), \dots\}$ where $\pi(u, T)$ is the probability of u belonging to topic T. We invert the topic-unit representation in Table 1, where each u is represented as a topic vector. The correlation $corr(.)$ between u_i, u_j is measured by cosine similarity $sim(.)$ on topic distribution vector $\vec{u}_i$, $\vec{u}_j$.

$$corr(u_i, u_j) = sim_{\text{topic}}(\vec{u}_i, \vec{u}_j) \quad (1)$$

Table 1: Inverted representation of topic-unit vector.

$\vec{u}_1$	$\pi(u_1, T_1)$	$\pi(u_1, T_2)$	$\dots$	$\pi(u_1, T_n)$
$\vec{u}_2$	$\pi(u_2, T_1)$	$\pi(u_2, T_2)$	$\dots$	$\pi(u_2, T_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vec{u}_{\|M\|}$	$\pi(u_{\|M\|}, T_1)$	$\pi(u_{\|M\|}, T_2)$	$\dots$	$\pi(u_{\|M\|}, T_n)$

When the summary is *hypertexified* by established linkage, users click into the generated summary to examine what they are interested in. A single click on one sentence become multiple clicks via click smoothing when the indicative function $I(u|q) = 1$.

$$I(u|q) = \begin{cases} 1 & q \text{ contains } u; \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The click smoothing brings pseudo clicks q' associated with u' and contexts c'. The entire user feedback texts $\mathcal{A}$ from q can be written as:

$$\mathcal{A}(q) = I(u|q) \sum_{j=1}^{|M|} corr(u', u)(u' + \gamma \cdot c') + \gamma \cdot c \quad (3)$$

where γ is the weight tradeoff between u and associated contexts c. If $I(u|q) = 0$, only the examined context c is feedbacked for user preference; otherwise, correlative contexts with u are taken into consideration, which is a process of impact diffusion.

3.2 Personalized Summarization

Traditional summarization involves two essential requirements: (1) **coverage**: the summary should keep alignment with the source collection, which is proved to be significant (Li et al., 2009). (2) **diversity**: according to MMR principle (Goldstein et al., 1999) and its applications (Wan et al., 2007b; Wan and Yang, 2008), a good summary should be concise and contain as few redundant sentences as possible, i.e., two sentences providing similar information should not both present. According to our

investigation, we observe that a well generated summary should properly consider a key component of (3) **user interests**, which captures user preference to summarize what they are interested in.

All above requirements involve a measurement of similarity between two word distributions Θ_1 and Θ_2. Cosine, Kullback-Leibler divergence D_{KL} and Jensen Shannon divergence D_{JS} are all able to measure the similarity, but (Louis and Nenkova, 2009) indicate the superiority of D_{JS} in summarization task. We also introduce a pair of decreasing/increasing logistic functions, $\mathcal{L}_1(x) = 1/(1 + e^x)$ and $\mathcal{L}_2(x) = e^x/(1 + e^x)$, to map the divergence into interval [0,1]. V is the vocabulary set and *tf* denotes the term frequency for word w.

$$D_{JS}(\Theta_1||\Theta_2) = \frac{1}{2}[D_{KL}(\Theta_1||\Theta_2)+D_{KL}(\Theta_2||\Theta_1)]$$

where

$$D_{KL}(\Theta_1||\Theta_2) = \sum_{k\in V} p(w|\Theta_1) log \frac{p(w|\Theta_1)}{p(w|\Theta_2)}$$

where

$$p(w|\Theta) = \frac{tf(w,\Theta)}{\sum_{w'} tf(w',\Theta)}.$$

Modeling Interest for User Utility. Given a generated summary S, users tend to scrutinize texts relevant to their interests. Texts related to user implicit feedback are collected as $\mathcal{A} = \sum_{i=1}^{|Q|} \mathcal{A}(q_i)$. Intuitively, the smaller distance between the word distribution of final summary (Θ_S) and the word distribution of user preference ($\Theta_{\mathcal{A}}$), the higher utility of user interests $\mathcal{U}_{user}(S)$ will be, i.e.,

$$\mathcal{U}_{user}(S) = \mathcal{L}_1(D_{JS}(\Theta_S||\Theta_{\mathcal{A}})). \quad (4)$$

We model the utility of traditional summarization $\mathcal{U}_{trad}(S)$ using a linear interpolation controlled by parameter δ between utility from *coverage* $\mathcal{U}_c(S)$ and utility $\mathcal{U}_d(S)$ from *diversity*:

$$\mathcal{U}_{trad}(S) = \mathcal{U}_c(S) + \delta \cdot \mathcal{U}_d(S). \quad (5)$$

Coverage Utility. The summary should share a closer word distribution with the source collection (Allan et al., 2001; Li et al., 2009). A good summary focuses on minimizing the loss of main information from the whole collection D. Utility from coverage $\mathcal{U}_c(S)$ is defined as follows and for coverage utility, smaller divergence is desired.

$$\mathcal{U}_c(S) = \mathcal{L}_1(D_{JS}(\Theta_S||\Theta_D)). \quad (6)$$

Diversity Utility. Diversity measures the novelty degree of any sentence s compared with all other sentences within S, i.e., the distances between all other sentences and itself. Diversity utility $\mathcal{U}_d(S)$ is an average novelty score for all sentences in S. For diversity utility, larger distance is desired, and hence we use the increasing function $\mathcal{L}_2$ as follows:

$$\mathcal{U}_d(S) = \frac{1}{|S|}\sum_{s\in S} \mathcal{L}_2(D_{JS}(\Theta_s||\Theta_{(S-s)})). \quad (7)$$

3.3 Balanced Optimization Framework

A well generated summary S should be sufficiently aligned with the original source corpus, and also be optimized given the user interests. The utility of an individual summary $\mathcal{U}(S)$ is evaluated by the weighted combination of these components, controlled by parameter λ for balanced weights.

$$\mathcal{U}(S) = \mathcal{U}_{trad}(S) + \lambda \cdot \mathcal{U}_{user}(S) \quad (8)$$

Given the sentence set D and the compression rate ϕ, there are $\phi \cdot |D|$ out of $|D|$ possibilities to generate S. The IPS task is to predict the optimized sentence subset of S^* from the space of all combinations. The objective function is as follows:

$$S^* = \underset{S}{\operatorname{argmax}}\, \mathcal{U}(S). \quad (9)$$

As $\mathcal{U}(S)$ is measured based on preferred interests from user interaction within a generation in our system, we extract S iteratively to approximate S^*, i.e, maximize $\mathcal{U}(S)$ based on the user feedbacks from the interaction sessions. Each session is an iteration. We use a similar framework as we have proposed in (Yan et al., 2011).

During every session, the top ranked sentences are strong candidates for the summary to generate and the rank methodology is based on the metrics $\mathcal{U}(.)$. The algorithm tends to highly rank sentences which are with both coverage utility and interest utility, and are diversified in balance: we rank each sentence s according to $\mathcal{U}(s)$ under such metrics.

Consider $S^{(n-1)}$ generated in the (*n-1*)-th session which consists of top $\phi|D|$ ranked sentences, as well

as the top $\phi|D|$ ranked sentences in the *n*-th iteration (denoted by $\mathcal{O}^{(n)}$), they have an intersection set of $\mathcal{Z}^{(n)} = S^{n-1} \cap \mathcal{O}^n$. There is a substitutable sentence set $\mathcal{X}^{(n)} = S^{(n-1)} - \mathcal{Z}^{(n)}$ and a new candidate sentence set $\mathcal{Y}^{(n)} = \mathcal{O}^{(n)} - \mathcal{Z}^{(n)}$. We substitute $\mathrm{x}^{(n)}$ sentences with $\mathrm{y}^{(n)}$, where $\mathrm{x}^{(n)} \subseteq \mathcal{X}^{(n)}$ and $\mathrm{y}^{(n)} \subseteq \mathcal{Y}^{(n)}$. During every iteration, our goal is to find a substitutive pair $<\mathrm{x}, \mathrm{y}>$ for S:

$$<\mathrm{x}, \mathrm{y}> : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}.$$

To measure the performance of such a substitution, a discriminant utility gain function $\Delta\mathcal{U}_{\mathrm{x},\mathrm{y}}$

$$\begin{aligned}\Delta\mathcal{U}^{(n)}_{\mathrm{x}^{(n)},\mathrm{y}^{(n)}} &= \mathcal{U}(S^{(n)}) - \mathcal{U}(S^{(n-1)}) \\ &= \mathcal{U}((S^{(n-1)} - \mathrm{x}^{(n)}) \cup \mathrm{y}^{(n)}) - \mathcal{U}(S^{(n-1)})\end{aligned} \tag{10}$$

is employed to quantify the penalty. Therefore, we predict the substitutive pair by maximizing the gain function $\Delta\mathcal{U}_{\mathrm{x},\mathrm{y}}$ over the state set $\mathcal{R}$, with a size of $\sum_{k=0}^{\mathcal{Y}} A^k_{\mathcal{X}} C^k_{\mathcal{Y}}$, where $<\mathrm{x}, \mathrm{y}> \in \mathcal{R}$. Finally the objective function of Equation (9) changes into maximization of utility gain by substitute $\hat{\mathrm{x}}$ with $\hat{\mathrm{y}}$ during each iteration:

$$< \hat{\mathrm{x}}, \hat{\mathrm{y}} >= \underset{\mathrm{x}\subseteq\mathcal{X},\mathrm{y}\subseteq\mathcal{Y}}{\operatorname{argmax}} \Delta\mathcal{U}_{\mathrm{x},\mathrm{y}}. \tag{11}$$

Note that the objectives of interest utility optimization and traditional utility optimization are not always the same because the word distributions in these texts are usually different. The substitutive pair $<\mathrm{x}, \mathrm{y}>$ may perform well based on the user preference component while not on the traditional summary part and vice versa. There is a tradeoff between both user optimization and traditional optimization and hence we need to balance them by λ.

The objective Equation (11) is actually to maximize $\Delta\mathcal{U}(S)$ from all possible substitutive pairs between two iteration sessions to generate S. The algorithm is shown in Algorithm 1. The threshold ϵ is set at 0.001 in this study.

Algorithm 1 Regenerative Optimization

Input: D, ϵ, ϕ
for all $s \in D$ **do**
  calculate $\mathcal{U}_{trad}(s)$
end for
$S \leftarrow$ top $\phi|D|$ ranked sentences
while new generation=TRUE **do**
  collect clicks and update utility from $\mathcal{U}'$ to $\mathcal{U}$
  if $|\mathcal{U}(S) - \mathcal{U}'(S)| > \epsilon$ **then**
    for all $s \in D$ **do**
      calculate $\mathcal{U}(s)$
    end for
    $\mathcal{O} \leftarrow$ top $\phi|D|$ ranked sentences by $\mathcal{U}(s)$
    $\mathcal{Z} \leftarrow S \cap \mathcal{O}$
    $\mathcal{X} \leftarrow S - \mathcal{Z}$, $\mathcal{Y} \leftarrow \mathcal{O} - \mathcal{Z}$
    for all $<\mathrm{x}, \mathrm{y}>$ pair where $\mathrm{x} \subseteq \mathcal{X}, \mathrm{y} \subseteq \mathcal{Y}$ **do**
      $\Delta\mathcal{U}_{\mathrm{x},\mathrm{y}} = \mathcal{U}((S - \mathrm{x}) \cup \mathrm{y}) - \mathcal{U}(S)$
    end for
    $< \hat{\mathrm{x}}, \hat{\mathrm{y}} >= \operatorname{argmax} \Delta\mathcal{U}_{\mathrm{x},\mathrm{y}}$
    $S \leftarrow (S - \hat{\mathrm{x}}) \cup \hat{\mathrm{y}}$
  end if
end while

4 Experiments and Evaluation

4.1 Datasets

IPS can be tested on any document set but a tiny corpus to summarize may not cover abundant effective interests to attract user clicks indicating their preference. Besides, the scenario of small corpus is not quite practical for the exponential growing web. Therefore, we test IPS on large real world datasets. We build 4 news story sets which consist of documents and reference summaries to evaluate our proposed framework empirically. We downloaded 5197 news articles from 10 selected sources. As shown in Table 2, three of the sources are in UK, one of them is in China and the rest are in US. We choose them because many of these websites provide handcrafted summaries for their special reports, which serve as reference summaries. These events belong to different categories of Rule of Interpretation (ROI) (Kumaran and Allan, 2004). Statistics are in Table 3.

4.2 Experimental System Setups

• **Preprocessing.** Given a collection of documents, we first decompose them into sentences. Stop-words are removed and words stemming is performed. Then the word distributions can be calculated.

• **User Interface Design.** Users are required to specify the overall compression rate ϕ and the system extracts $\phi|D|$ sentences according to user utility

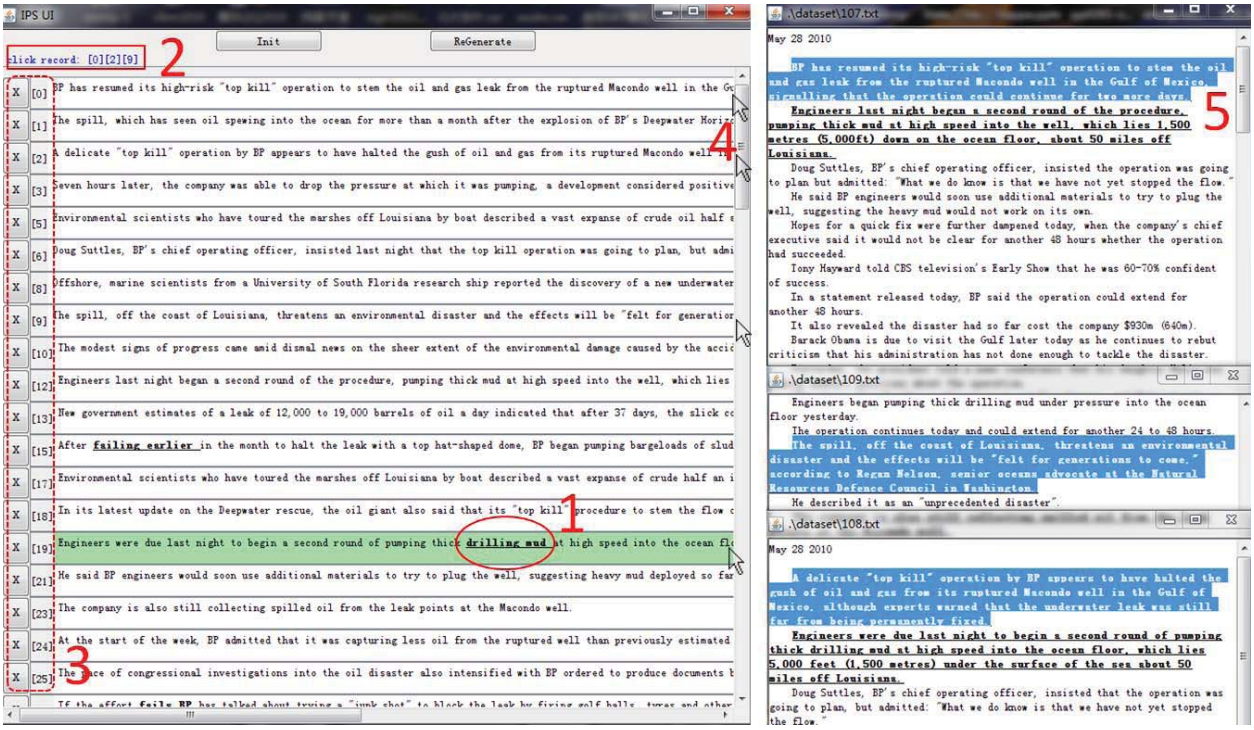


Figure 1: A demonstration system for Interactive Personalized Summarization when compression rate ϕ is specified (e.g. 5%). For convenience of browsing, we number the selected sentences (see in part 3). Extracted semantic units, such as "drilling mud", are in bold and underlined format (see in part 1). When the user clicks a sentence (part 4), the clicked sentence **ID** is kept in the *click record* (part 2). Mis-clicked records revocation can be operated by clicking the deletion icon "X" (see in part 3). Once a sentence is clicked, user can track the sentence into the popup source document to examine the contexts. The selected sentences are highlighted in the source documents (see in part 5).

Table 2: News sources of 4 datasets

News Sources	Nation	News Sources	Nation
BBC	UK	Fox News	US
Xinhua	China	MSNBC	US
CNN	US	Guardian	UK
ABC	US	New York Times	US
Reuters	UK	Washington Post	US

Table 3: Detailed basic information of 4 datasets.

News Subjects	#size	#docs	#RS	Avg.L
1.Influenza A	115026	2557	5	83
2.BP Oil Spill	63021	1468	6	76
3.Haiti Earthquake	12073	247	2	32
4.Jackson Death	37819	925	3	64

#size: total sentence counts; #RS: the number of reference summaries; Avg.L: average length of reference summary measured in sentences.

and traditional utility. User utility is obtained from interaction. The system keeps the clicked sentence records and calculates the user feedback by Equation (3) during every session. Consider sometimes users click into the summary due to confusion or mis-operations, but not their real interests. The system supports click records revocation. More details of the user interface is demonstrated in Figure 1.

4.3 Evaluation Metrics

We include both subjective evaluation from 3 evaluators based on their personalized interests and preference, and the objective evaluation based on the widely used ROUGE metrics (Lin and Hovy, 2003).

Evaluator Judgments

Evaluators are requested to express an opinion over all summaries based on the sentences which they deem to be important for the news. In general a summary can be rated in a 5-point scale, where "1" for "terrible", "2" for "bad", "3" for "normal", "4" for "good" and "5" for "excellent". Evaluators are allowed to judge at any scores between 1 and 5, e.g. a score of "3.3" is adopted when the evaluator feels difficult to decide whether "3" or "4" is more

appropriate but with preference towards "3".

ROUGE Evaluation

The DUC usually officially employs ROUGE measures for summarization evaluation, which measures summarization quality by counting overlapping units such as the N-gram, word sequences, and word pairs between the candidate summary and the reference summary. We use ROUGE-N as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in \{\text{RefSum}\}} \sum_{\text{N-gram} \in S} \text{Count}_{\text{match}}(\text{N-gram})}{\sum_{S \in \{\text{RefSum}\}} \sum_{\text{N-gram} \in S} \text{Count (N-gram)}}$$

where N stands for the length of the N-gram and N-gram$\in$RefSum denotes the N-grams in the reference summaries while N-gram$\in$CandSum denotes the N-grams in the candidate summaries. $\text{Count}_{\text{match}}$(N-gram) is the maximum number of N-gram in the candidate summary and in the set of reference summaries. $\text{Count}_{\text{(N-gram)}}$ is the number of N-grams in the reference summaries or candidate summary.

According to (Lin and Hovy, 2003), among all sub-metrics in ROUGE, ROUGE-N (N=1, 2) is relatively simple and works well. In this paper, we evaluate our experiments using all methods provided by the ROUGE package (version 1.55) and only report ROUGE-1, since the conclusions drawn from different methods are quite similar. Intuitively, the higher the ROUGE scores, the similar two summaries are.

4.4 Algorithms for Comparison

We implement the following widely used multi-document summarization algorithms as the baseline systems, which are all designed for traditional summarization without user interaction. For fairness we conduct the same preprocessing for all algorithms.

Random: The method selects sentences randomly for each document collection.

Centroid: The method applies MEAD algorithm (Radev et al., 2004) to extract sentences according to the following parameters: centroid value, positional value, and first-sentence overlap.

GMDS: The Graph-based MDS proposed by (Wan and Yang, 2008) first constructs a sentence connectivity graph based on cosine similarity and then selects important sentences based on the concept of eigenvector centrality.

IPS_{ini}: The initial generated summary from IPS merely models *coverage* and *diversity* utility, which is similar to the previous work described in (Allan et al., 2001) with different goals and frameworks.

IPS: Our proposed algorithms with personalization component to capture *interest* by user feedbacks. IPS generates summaries via iterative sentence substitutions within user interactive sessions.

RefSum: As we have used multiple reference summaries from websites, we not only provide ROUGE evaluations of the competing systems but also of the reference summaries against each other, which provides a good indicator of not only the upper bound ROUGE score that any system could achieve, but also human inconsistency among reference summaries, indicating personalization.

4.5 Overall Performance Comparison

We take the average ROUGE-1 performance and human ratings on all sets. The overall results are shown in Figure 2 and details are listed in Tables 4$\sim$6.

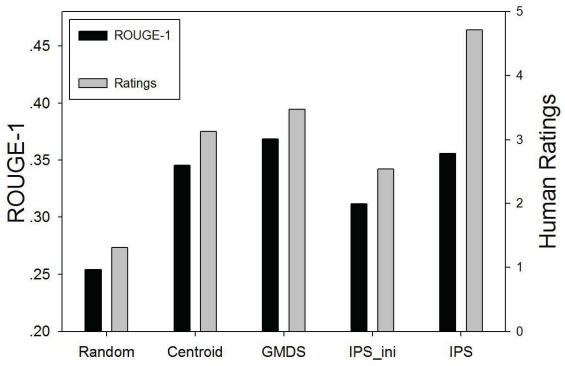


Figure 2: Overall performance on 6 datasets.

From the results, we have following observations:

• Random has the worst performance as expected, both in ROUGE-1 scores and human judgements.

• The ROUGE-1 and human ratings of Centroid and GMDS are better than those of Random. This is mainly because the Centroid based algorithm takes into account positional value and first-sentence overlap, which facilitates main aspects summarization and PageRank-based GMDS ranks the sentence using eigenvector centrality which implicitly accounts for information subsumption among all sentences.

• In general, the GMDS system slightly outperforms Centroid system in ROUGE-1, but the human judgements of GMDS and Centroid are of no significant difference. This is probably due to the difficulty

Table 4: Overall performance comparison on *Influenza A*. ROI* category: Science.

Systems	R-1	95%-conf.	H-1	H-2	H-3
RefSum	0.491	0.44958	3.5	3.0	3.9
Random	0.257	0.75694	1.2	1.0	1.0
Centroid	0.331	0.45073	2.5	3.0	3.5
GMDS	**0.364**	0.33269	3.0	2.7	3.5
IPS_{ini}	0.302	0.21213	2.0	2.5	2.5
IPS	0.337	0.46757	**4.8**	**4.5**	**4.5**

Table 5: Overall performance comparison on *BP Oil Leak*. ROI category: Accidents.

Systems	R-1	95%-conf.	H-1	H-2	H-3
RefSum	0.517	0.48618	4.0	3.3	3.9
Random	0.262	0.64406	1.5	1.0	1.5
Centroid	0.369	0.34743	3.2	3.0	3.5
GMDS	**0.389**	0.43877	3.5	3.0	3.9
IPS_{ini}	0.327	0.53722	3.0	2.5	3.0
IPS	0.372	0.35681	**4.8**	**4.5**	**4.5**

Table 6: Overall performance comparison on *Haiti Earthquake*. ROI category: Disasters.

Systems	R-1	95%-conf.	H-1	H-2	H-3
RefSum	0.528	0.30450	3.8	4.0	4.0
Random	0.266	0.75694	1.5	1.5	1.8
Centroid	0.362	0.43045	3.6	3.0	4.0
GMDS	0.380	0.33694	3.9	3.5	4.0
IPS_{ini}	0.331	0.34120	2.8	2.5	3.0
IPS	**0.391**	0.40069	**5.0**	**4.7**	**5.0**

Table 7: Overall performance comparison on *Michael Jackson Death*. ROI category: Legal Cases.

Systems	R-1	95%-conf.	H-1	H-2	H-3
RefSum	0.482	0.47052	3.5	3.5	4.0
Random	0.232	0.52426	1.2	1.0	1.5
Centroid	0.320	0.21045	3.0	2.5	2.7
GMDS	**0.341**	0.30070	3.5	3.3	3.9
IPS_{ini}	0.287	0.48526	2.5	2.0	2.2
IPS	0.324	0.36897	**5.0**	**4.5**	**4.8**

*ROI: news categorization defined by Linguistic Data Consortium. Available at http://www.ldc.upenn.edu/projects/tdt4/annotation

of human judgements on comparable summaries.

• The results of ROUGE-1 and ratings for IPS_{ini} are better than Random but worse than Centroid and GMDS. The reason in this case may be that IPS_{ini} does not capture sufficient attributes: coverage and diversity are merely fundamental requirements.

• Traditional summarization considers sentence selection based on corpus only, and hence neglects user interests. Many sentences are extracted due to arbitrary assumption of reader preference, which results in a low user satisfaction. Human judgements under our proposed IPS framework greatly outperform baselines, indicating that the appropriate use of human interests for summarization are beneficial.

Table 8: Ratings consistency between evaluators: mean ± standard deviation over the 4 datasets.

RefSum	Evaluator 1	Evaluator 2	Evaluator 3
Evaluator 1		0.35±0.09	0.30±0.33
Evaluator 2			0.50±0.14

Random	Evaluator 1	Evaluator 2	Evaluator 3
Evaluator 1		0.23±0.04	0.20±0.02
Evaluator 2			0.33±0.06

Centroid	Evaluator 1	Evaluator 2	Evaluator 3
Evaluator 1		0.45±0.03	0.50±0.12
Evaluator 2			0.55±0.11

GMDS	Evaluator 1	Evaluator 2	Evaluator 3
Evaluator 1		0.35±0.02	0.35±0.03
Evaluator 2			0.70±0.03

IPS_{ini}	Evaluator 1	Evaluator 2	Evaluator 3
Evaluator 1		0.45±0.01	0.25±0.04
Evaluator 2			0.30±0.06

IPS	Evaluator 1	Evaluator 2	Evaluator 3
Evaluator 1		0.35±0.01	0.18±0.02
Evaluator 2			0.28±0.04

The ROUGE-1 performance for IPS is not as ideal as that of GMDS. This situation may result from the divergence between user interests and general information provided by mass media propaganda, which again motivates the need for personalization.

Although the high disparities between different human evaluators have been observed in (Gong and Liu, 2001), we still examine the consistency among 3 evaluators and their preferred summaries to prove the motivation of personalization in our work.

4.6 Consistency Analysis for Personalization

The low ROUGE-1 scores of RefSum indicate the inconsistency among reference summaries. We conduct personalization analysis from two perspectives: (1) human rating consistency and (2) content consistency among human supervised summaries.

We calculate the mean and variance of rating variations among evaluator judgements, listed in Table

Table 9: Content consistency among evaluators supervised summaries.

	Evaluator 1	Evaluator 2	Evaluator 3
Evaluator 1		0.273	0.398
Evaluator 2	0.289		0.257
Evaluator 3	0.407	0.235	
RefSum	0.365	0.302	0.394

8. We see that for Random the average rating variation is 0.25, for IPS is 0.27, for IPS_{ini} is 0.33, for RefSum is 0.38, for GMDS is 0.47 and for Centroid is the highest, 0.50. Such phenomenon indicates for poor generated summaries, such as Random or IPS_{ini}, humans have consensus, but for normal summaries without personalized interests, they are likely to have disparities, surprisingly, even for RefSum. General summaries provided by mass media satisfy part of audiences, but obviously not all of them.

The high rating consistency of IPS indicates people tend to favor summaries generated according to their interests. We next examine content consistency of these summaries with high rating consistency.

As shown in Table 9, although highly scored, these human supervised summaries still have low content consistency (especially Evaluator 2). The low content consistency between RefSum and supervised summaries shows reader have individual personalization. Note that the inconsistency among evaluators is larger than that between RefSum and supervised summaries, indicating *interests* take a high proportion in evaluator supervised summaries.

4.7 Parameter Settings

δ controls coverage/diversity tradeoff. We tune δ on IPS_{ini} and apply the optimal δ directly in IPS. According to the statistics in (Yan et al., 2010), the semantic coherent context is about 7 sentences. Therefore, we empirically choose k=3 for the examined context window. The number of topics is set at n=50. We assign an equal weight ($\gamma = 1$) to semantic units and examined contexts according to analogical research of summarization from implicit feedbacks via clickthrough data (Sun et al., 2005).

λ is the key parameter in IPS approach, controlling the weight of user utility during the process of interactive personalized summarization.

Through Figure 3, we see that when λ is small

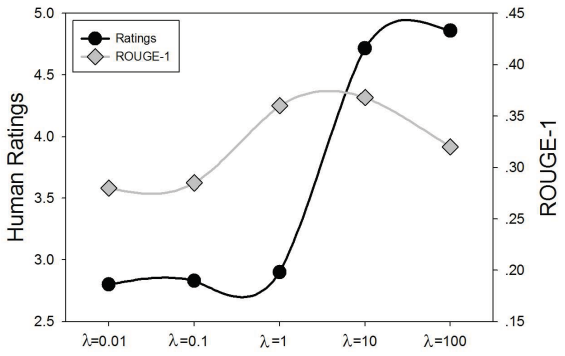


Figure 3: λ v.s. human ratings and ROUGE scores.

($\lambda \in [0.01, 0.1]$), both human judgements and ROUGE evaluation scores have little difference. When $\lambda \in [0.1, 1]$, ROUGE scores increase significantly but human satisfaction shows little response. $\lambda \in [1, 10]$ brings large user utility enhancement because user may find what they are interested in but ROUGE scores start to decay. When $\lambda \in [10, 100]$, ROUGE scores drop much because the emphasized user interests may guide the generated summaries divergent away from the original corpus.

In Figure 4 we examine how λ attracts user clicks and regeneration counts until satisfaction. As the result indicates, both counts increase as λ increases. When λ is small (from 0.01 to 0.1), readers find no more interesting aspects through clicks and regenerations and stop due to the bad user experience. As λ increases, the system mines more relevant sentences according to personalized interests and hence attracts user clicks and intention to regenerate.

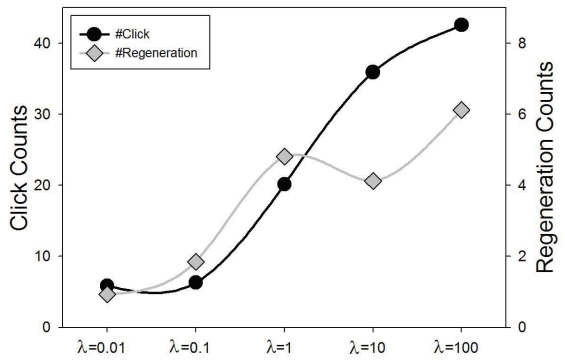


Figure 4: λ v.s. click counts and regeneration counts.

5 Conclusion

We present an important and novel summarization problem, Interactive Personalized Summarization (IPS), which generates summaries based on human−system interaction for "interests" and personalization. We formally formulate IPS as a combination of user utility and traditional summary utility, such as coverage and diversity. We implement a system under such framework for experiments on real web datasets to compare all approaches. Through our experiments we notice that user personalization of interests plays an important role in summary generation, which largely increase human ratings due to user satisfaction. Besides, our experiments indicate the inconsistency between user preferred summaries and reference summaries measured by ROUGE, and hence prove the effectiveness of personalization.

Acknowledgments

This work was partially supported by HGJ 2010 Grant 2011ZX01042-001-001 and NSFC with Grant No.61073082, 60933004. Rui Yan was supported by the MediaTek Fellowship.

References

James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal summaries of new topics. In *Proceedings of the 24th annual international SIGIR'01*, pages 10–18.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

G. Erkan and D.R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP'04*, volume 4.

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. 1999. Summarizing text documents: sentence selection and evaluation metrics. In *Proceedings of SIGIR'99*, pages 121–128.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th international ACM SIGIR conference*, SIGIR '01, pages 19–25.

Q. Guo and E. Agichtein. 2010. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *Proceeding of the 33rd international ACM SIGIR conference*, SIGIR'10, pages 130–137.

Giridhar Kumaran and James Allan. 2004. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR'04*, pages 297–304.

Anton Leuski, Chin-Yew Lin, and Eduard Hovy. 2003. ineats: interactive multi-document summarization. In *Proceedings of ACL'03*, pages 125–128.

Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of WWW'09*, pages 71–80.

Chin-Yew Lin and Eduard Hovy. 2002. From single to multi-document summarization: a prototype system and its evaluation. In *Proceedings of ACL'02*, pages 457–464.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of NAACL'03*, pages 71–78.

Annie Louis and Ani Nenkova. 2009. Automatically evaluating content selection in summarization without human models. In *EMNLP'09*, pages 306–314.

R. Mihalcea and P. Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP*, volume 5.

D.R. Radev, H. Jing, and M. Sty. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40(6):919–938.

Jian-Tao Sun, Dou Shen, Hua-Jun Zeng, Qiang Yang, Yuchang Lu, and Zheng Chen. 2005. Web-page summarization using clickthrough data. In *Proceedings of SIGIR'05*, pages 194–201.

Stephen Wan and Cécile Paris. 2008. In-browser summarisation: generating elaborative summaries biased towards the reading context. In *ACL-HLT'08*, pages 129–132.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of SIGIR'08*, pages 299–306.

X. Wan, J. Yang, and J. Xiao. 2007a. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of IJCAI*, volume 7, pages 2903–2908.

X. Wan, J. Yang, and J. Xiao. 2007b. Single document summarization with document expansion. In *Proceedings of the 22nd AAAI'07*, pages 931–936.

Rui Yan, Yu Li, Yan Zhang, and Xiaoming Li. 2010. Event recognition from news webpages through latent ingredients extraction. In *AIRS'10*, pages 490–501.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *Proceedings of the 34th annual international ACM SIGIR'11*.

Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achanauparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical Keyphrase Extraction from Twitter. In *Proceedings of ACL-HLT'11*.

Tuning as Ranking

Mark Hopkins and Jonathan May
SDL Language Weaver
Los Angeles, CA 90045
{mhopkins,jmay}@sdl.com

Abstract

We offer a simple, effective, and scalable method for statistical machine translation parameter tuning based on the pairwise approach to ranking (Herbrich et al., 1999). Unlike the popular MERT algorithm (Och, 2003), our pairwise ranking optimization (PRO) method is not limited to a handful of parameters and can easily handle systems with thousands of features. Moreover, unlike recent approaches built upon the MIRA algorithm of Crammer and Singer (2003) (Watanabe et al., 2007; Chiang et al., 2008b), PRO is easy to implement. It uses off-the-shelf linear binary classifier software and can be built on top of an existing MERT framework in a matter of hours. We establish PRO's scalability and effectiveness by comparing it to MERT and MIRA and demonstrate parity on both phrase-based and syntax-based systems in a variety of language pairs, using large scale data scenarios.

1 Introduction

The MERT algorithm (Och, 2003) is currently the most popular way to tune the parameters of a statistical machine translation (MT) system. MERT is well-understood, easy to implement, and runs quickly, but can behave erratically and does not scale beyond a handful of features. This lack of scalability is a significant weakness, as it inhibits systems from using more than a couple dozen features to discriminate between candidate translations and stymies feature development innovation.

Several researchers have attempted to address this weakness. Recently, Watanabe et al. (2007) and Chiang et al. (2008b) have developed tuning methods using the MIRA algorithm (Crammer and Singer, 2003) as a nucleus. The MIRA technique of Chiang et al. has been shown to perform well on large-scale tasks with hundreds or thousands of features (2009). However, the technique is complex and architecturally quite different from MERT. Tellingly, in the entire proceedings of ACL 2010 (Hajič et al., 2010), only one paper describing a statistical MT system cited the use of MIRA for tuning (Chiang, 2010), while 15 used MERT.[1]

Here we propose a simpler approach to tuning that scales similarly to high-dimensional feature spaces. We cast tuning as a ranking problem (Chen et al., 2009), where the explicit goal is to learn to correctly rank candidate translations. Specifically, we follow the *pairwise approach* to ranking (Herbrich et al., 1999; Freund et al., 2003; Burges et al., 2005; Cao et al., 2007), in which the ranking problem is reduced to the binary classification task of deciding between candidate translation pairs.

Of primary concern to us is the ease of adoption of our proposed technique. Because of this, we adhere as closely as possible to the established MERT architecture and use freely available machine learning software. The end result is a technique that scales and performs just as well as MIRA-based tuning, but which can be implemented in a couple of hours by anyone with an existing MERT implementation. Mindful that many would-be enhancements to the

[1] The remainder either did not specify their tuning method (though a number of these used the Moses toolkit (Koehn et al., 2007), which uses MERT for tuning) or, in one case, set weights by hand.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1352–1362,
Edinburgh, Scotland, UK, July 27–31, 2011.

state-of-the-art are false positives that only show improvement in a narrowly defined setting or with limited data, we validate our claims on both syntax and phrase-based systems, using multiple language pairs and large data sets.

We describe tuning in abstract and somewhat formal terms in Section 2, describe the MERT algorithm in the context of those terms and illustrate its scalability issues via a synthetic experiment in Section 3, introduce our pairwise ranking optimization method in Section 4, present numerous large-scale MT experiments to validate our claims in Section 5, discuss some related work in Section 6, and conclude in Section 7.

2 Tuning

In Figure 1, we show an example *candidate space*, defined as a tuple $\langle \Delta, I, J, f, e, \mathbf{x} \rangle$ where:

- Δ is a positive integer referred to as the *dimensionality* of the space
- I is a (possibly infinite) set of positive integers, referred to as *sentence indices*
- J maps each sentence index to a (possibly infinite) set of positive integers, referred to as *candidate indices*
- f maps each sentence index to a sentence from the source language
- e maps each pair $\langle i, j \rangle \in I \times J(i)$ to the j^{th} target-language *candidate translation* of source sentence $f(i)$
- $\mathbf{x}$ maps each pair $\langle i, j \rangle \in I \times J(i)$ to a Δ-dimension *feature vector* representation of $e(i, j)$

The example candidate space has two source sentences, three candidate translations for each source sentence, and feature vectors of dimension 2. It is an example of a *finite* candidate space, defined as a candidate space for which I is finite and J maps each index of I to a finite set.

A *policy* of candidate space $\langle \Delta, I, J, f, e, \mathbf{x} \rangle$ is a function that maps each member $i \in I$ to a member of $J(i)$. A policy corresponds to a choice of one candidate translation for each source sentence. For the example in Figure 1, policy $p_1 = \{1 \mapsto 2, 2 \mapsto 3\}$ corresponds to the choice of "he does not go" for the first source sentence and "I do not go" for the second source sentence. Obviously some policies are better than others. Policy $p_2 = \{1 \mapsto 3, 2 \mapsto 1\}$ corresponds to the inferior translations "she not go" and "I go not."

We assume the MT system distinguishes between policies using a scoring function for candidate translations of the form $h_{\mathbf{w}}(i, j) = \mathbf{w} \cdot \mathbf{x}(i, j)$, where $\mathbf{w}$ is a weight vector of the same dimension as feature vector $\mathbf{x}(i, j)$. This scoring function extends to a policy p by summing the cost of each of the policy's candidate translations: $H_{\mathbf{w}}(p) = \sum_{i \in I} h_{\mathbf{w}}(i, p(i))$. As can be seen in Figure 1, using $\mathbf{w} = [-2, 1]$, $H_{\mathbf{w}}(p_1) = 9$ and $H_{\mathbf{w}}(p_2) = -8$.

The goal of tuning is to learn a weight vector $\mathbf{w}$ such that $H_{\mathbf{w}}(p)$ assigns a high score to good policies, and a low score to bad policies.[2] To do so, we need information about which policies are good and which are bad. This information is provided by a "gold" scoring function G that maps each policy to a real-valued score. Typically this gold function is BLEU (Papineni et al., 2002), though there are several common alternatives (Lavie and Denkowski, 2009; Melamed et al., 2003; Snover et al., 2006; Chiang et al., 2008a).

We want to find a weight vector $\mathbf{w}$ such that $H_{\mathbf{w}}$ behaves "similarly" to G on a candidate space s. We assume a loss function $l_s(H_{\mathbf{w}}, G)$ which returns the real-valued loss of using scoring function $H_{\mathbf{w}}$ when the gold scoring function is G and the candidate space is s. Thus, we may say the goal of tuning is to find the weight vector $\mathbf{w}$ that minimizes loss.

3 MERT

In general, the candidate space may have infinitely many source sentences, as well as infinitely many candidate translations per source sentence. In practice, tuning optimizes over a finite subset of source sentences[3] and a finite subset of candidate translations as well. The classic tuning architecture used in the dominant MERT approach (Och, 2003) forms the translation subset and learns weight vector $\mathbf{w}$ via

[2] Without loss of generality, we assume that a higher score indicates a better translation.

[3] See Section 5.2 for the tune sets used in this paper's experiments.

Source Sentence		Candidate Translations				
i	$f(i)$	j	$e(i,j)$	$\mathbf{x}(i,j)$	$h_{\mathbf{w}}(i,j)$	$g(i,j)$
1	"il ne va pas"	1	"he goes not"	[2 4]	0	0.28
		2	"he does not go"	[3 8]	2	0.42
		3	"she not go"	[6 1]	-11	0.12
2	"je ne vais pas"	1	"I go not"	[-3 -3]	3	0.15
		2	"we do not go"	[1 -5]	-7	0.18
		3	"I do not go"	[-5 -3]	7	0.34

Figure 1: Example candidate space of dimensionality 2. Note: $I = \{1, 2\}$, $J(1) = J(2) = \{1, 2, 3\}$. We also show a local scoring function $h_{\mathbf{w}}(i, j)$ (where $\mathbf{w} = [-2, 1]$) and a local gold scoring function $g(i, j)$.

Algorithm TUNE(s, G):

1: **initialize pool:** let $s' = \langle \Delta, I', J', f, e, \mathbf{x} \rangle$, where $I' \subseteq I$ and $J' = \emptyset$
2: **for** the desired number of iterations **do**
3: **candidate generation:** choose index pairs (i, j); for each, add j to $J'(i)$
4: **optimization:** find vector $\mathbf{w}$ that minimizes $l_{s'}(H_{\mathbf{w}}, G)$
5: **return** $\mathbf{w}$

Figure 2: Schema for iterative tuning of base candidate space $s = \langle \Delta, I, J, f, e, \mathbf{x} \rangle$ w.r.t. gold function G.

a feedback loop consisting of two phases. Figure 2 shows the pseudocode. During *candidate generation*, candidate translations are selected from a base candidate space s and added to a finite candidate space s' called the *candidate pool*. During *optimization*, the weight vector $\mathbf{w}$ is optimized to minimize loss $l_{s'}(H_{\mathbf{w}}, G)$.

For its candidate generation phase, MERT generates the k-best candidate translations for each source sentence according to $h_{\mathbf{w}}$, where $\mathbf{w}$ is the weight vector from the previous optimization phase (or an arbitrary weight vector for the first iteration).

For its optimization phase, MERT defines the loss function as follows:

$$l_s(H_{\mathbf{w}}, G) = \max_p G(p) - G(\arg\max_p H_{\mathbf{w}}(p))$$

In other words, it prefers weight vectors $\mathbf{w}$ such that the gold function G scores $H_{\mathbf{w}}$'s best policy as highly as possible (if $H_{\mathbf{w}}$'s best policy is the same as G's best policy, then there is zero loss). Typically the optimization phase is implemented using Och's line optimization algorithm (2003).

MERT has proven itself effective at tuning candidate spaces with low dimensionality. However, it is often claimed that MERT does not scale well with dimensionality. To test this claim, we devised the following synthetic data experiment:

1. We created a gold scoring function G that is *also* a linear function of the same form as $H_{\mathbf{w}}$, i.e., $G(p) = H_{\mathbf{w}^*}(p)$ for some gold weight vector $\mathbf{w}^*$. Under this assumption, the role of the optimization phase reduces to learning back the gold weight vector $\mathbf{w}^*$.

2. We generated a Δ-dimensionality candidate pool with 500 source "sentences" and 100 candidate "translations" per sentence. We created the corresponding feature vectors by drawing Δ random real numbers uniformly from the interval $[0, 500]$.

3. We ran MERT's line optimization on this synthetic candidate pool and compared the learned weight vector $\mathbf{w}$ to the gold weight vector $\mathbf{w}^*$ using cosine similarity.

We used line optimization in the standard way, by generating 20 random starting weight vectors and hill-climbing on each independently until no further progress is made, then choosing the final weight vector that minimizes loss. We tried various dimensionalities from 10 to 1000. We repeated each setting three times, generating different random data each time. The results in Figure 3 indicate that as the dimensionality of the problem increases MERT rapidly loses the ability to learn $\mathbf{w}^*$. Note that this synthetic problem is considerably easier than a real MT scenario, where the data is noisy and interdependent, and the gold scoring function is nonlinear. If

MERT cannot scale in this simple scenario, it has little hope of succeeding in a high-dimensionality deployment scenario.

4 Optimization via Pairwise Ranking

We would like to modify MERT so that it scales well to high-dimensionality candidate spaces. The most prominent example of a tuning method that performs well on high-dimensionality candidate spaces is the MIRA-based approach used by Watanabe et al. (2007) and Chiang et al. (2008b; 2009). Unfortunately, this approach requires a complex architecture that diverges significantly from the MERT approach, and consequently has not been widely adopted. Our goal is to achieve the same performance with minimal modification to MERT.

With MERT as a starting point, we have a choice: modify candidate generation, optimization, or both. Although alternative candidate generation methods have been proposed (Macherey et al., 2008; Chiang et al., 2008b; Chatterjee and Cancedda, 2010), we will restrict ourselves to MERT-style candidate generation, in order to minimize divergence from the established MERT tuning architecture. Instead, we focus on the optimization phase.

4.1 Basic Approach

While intuitive, the MERT optimization module focuses attention on $H_{\mathbf{w}}$'s best policy, and not on its overall prowess at ranking policies. We will create an optimization module that directly addresses $H_{\mathbf{w}}$'s ability to rank policies in the hope that this more holistic approach will generalize better to unseen data.

Assume that the gold scoring function G decomposes in the following way:

$$G(p) = \sum_{i \in I} g(i, p(i)) \tag{1}$$

where $g(i, j)$ is a local scoring function that scores the single candidate translation $e(i, j)$. We show an example g in Figure 1. For an arbitrary pair of candidate translations $e(i, j)$ and $e(i, j')$, the local gold function g tells us which is the better translation. Note that this induces a ranking on the candidate translations for each source sentence.

We follow the *pairwise approach* to ranking (Herbrich et al., 1999; Freund et al., 2003; Burges et al., 2005; Cao et al., 2007). In the pairwise approach, the learning task is framed as the classification of candidate pairs into two categories: correctly ordered and incorrectly ordered. Specifically, for candidate translation pair $e(i, j)$ and $e(i, j')$, we want: $g(i, j) > g(i, j') \Leftrightarrow h_{\mathbf{w}}(i, j) > h_{\mathbf{w}}(i, j')$. We can re-express this condition:

$$\begin{aligned} g(i, j) > g(i, j') &\Leftrightarrow h_{\mathbf{w}}(i, j) > h_{\mathbf{w}}(i, j') \\ &\Leftrightarrow h_{\mathbf{w}}(i, j) - h_{\mathbf{w}}(i, j') > 0 \\ &\Leftrightarrow \mathbf{w} \cdot \mathbf{x}(i, j) - \mathbf{w} \cdot \mathbf{x}(i, j') > 0 \\ &\Leftrightarrow \mathbf{w} \cdot (\mathbf{x}(i, j) - \mathbf{x}(i, j')) > 0 \end{aligned}$$

Thus optimization reduces to a classic binary classification problem. We create a labeled training instance for this problem by computing difference vector $\mathbf{x}(i, j) - \mathbf{x}(i, j')$, and labeling it as a positive or negative instance based on whether, respectively, the first or second vector is superior according to gold function g. To ensure balance, we consider both possible difference vectors from a pair. For example, given the candidate space of Figure 1, since $g(1, 1) > g(1, 3)$, we would add $([-4, 3], +)$ and $([4, -3], -)$ to our training set. We can then feed this training data directly to any off-the-shelf classification tool that returns a linear classifier, in order to obtain a weight vector $\mathbf{w}$ that optimizes the above condition. This weight vector can then be used directly by the MT system in the subsequent candidate generation phase. The exact loss function $l_{s'}(H_{\mathbf{w}}, G)$ optimized depends on the choice of classifier.[4]

Typical approaches to pairwise ranking enumerate all difference vectors as training data. For tuning however, this means $O(|I| * J_{max}^2)$ vectors, where J_{max} is the cardinality of the largest $J(i)$. Since I and J_{max} commonly range in the thousands, a full enumeration would produce billions of feature vectors. Out of tractability considerations, we sample from the space of difference vectors, using the sampler template in Figure 4. For each source sentence i, the sampler generates Γ candidate translation pairs $\langle j, j' \rangle$, and accepts each pair with probability $\alpha_i(|g(i, j) - g(i, j')|)$. Among the accepted pairs, it keeps the Ξ with greatest g differential, and adds their difference vectors to the training data.[5]

[4] See (Chen et al., 2009) for a brief survey.

[5] The intuition for biasing toward high score differential is

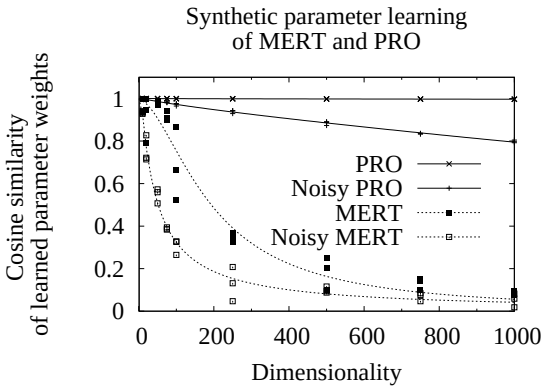


Figure 3: Result of synthetic data learning experiment for MERT and PRO, with and without added noise. As the dimensionality increases MERT is unable to learn the original weights but PRO still performs adequately.

Algorithm SAMPLER$_{s,g}$(Γ, Ξ, i, α_i):

1: $V = \langle\rangle$
2: **for** Γ samplings **do**
3: Choose $\langle j, j' \rangle \in J(i) \times J(i)$ uniformly at random.
4: With probability $\alpha_i(|g(i,j)\text{-}g(i,j')|)$, add $(\mathbf{x}(i,j), \mathbf{x}(i,j'), |g(i,j)\text{-}g(i,j')|)$ to V.
5: Sort V decreasingly by $|g(i,j)\text{-}g(i,j')|$.
6: **return** $(\mathbf{x}(i,j) - \mathbf{x}(i,j'), \mathsf{sign}(g(i,j)\text{-}g(i,j'))$ and $(\mathbf{x}(i,j')\text{-}\mathbf{x}(i,j), \mathsf{sign}(g(i,j')\text{-}g(i,j)))$ for each of the first Ξ members of V.

Figure 4: Pseudocode for our sampler. Arguments: $s = \langle \Delta, I, J, f, e, \mathbf{x} \rangle$ is a finite candidate space; g is a scoring function; Γ, Ξ, i are nonnegative integers; α_i is a function from the nonnegative real numbers to the real interval $[0, 1]$.

4.2 Scalability

We repeated the scalability study from Section 3, now using our pairwise ranking optimization (hereafter, PRO) approach. Throughout all experiments with PRO we choose $\Gamma = 5000$, $\Xi = 50$, and the following step function α for each α_i: [6]

$$\alpha(n) = \begin{cases} 0 \text{ if } n < 0.05 \\ 1 \text{ otherwise} \end{cases}$$

We used MegaM (Daumé III, 2004) as a binary classifier in our contrasting synthetic experiment and ran it "out of the box," i.e., with all default settings for binary classification.[7] Figure 3 shows that PRO is able to learn $\mathbf{w}^*$ nearly perfectly at all dimensionalities from 10 to 1000.

As noted previously, though, this is a rather simple task. To encourage a disconnect between g and $h_{\mathbf{w}}$ and make the synthetic scenario look more like MT reality, we repeated the synthetic experiments but added noise to each feature vector, drawn from a zero-mean Gaussian with a standard deviation of 500. The results of the noisy synthetic experiments, also in Figure 3 (the lines labeled "Noisy"), show that the pairwise ranking approach is less successful than before at learning $\mathbf{w}^*$ at high dimensionality, but still greatly outperforms MERT.

4.3 Discussion

The idea of learning from difference vectors also lies at the heart of the MIRA-based approaches (Watanabe et al., 2007; Chiang et al., 2008b) and the approach of Roth et al. (2010), which, similar to our method, uses sampling to select vectors. Here, we isolate these aspects of those approaches to create a simpler tuning technique that closely mirrors the ubiquitous MERT architecture. Among other simplifications, we abstract away the choice of MIRA as the classification method (our approach can use any classification technique that learns a separating hyperplane), and we eliminate the need for oracle translations.

An important observation is that BLEU does not satisfy the decomposability assumption of Equation (1). An advantage of MERT is that it can directly optimize for non-decomposable scoring functions like BLEU. In our experiments, we use the BLEU+1 approximation to BLEU (Liang et al., 2006) to determine class labels. We will nevertheless use BLEU to evaluate the trained systems.

that our primary goal is to ensure good translations are preferred to bad translations, and not to tease apart small differences.

[6] We obtained these parameters by trial-and-error experimentation on a single MT system (Urdu-English SBMT), then held them fixed throughout our experiments. We obtained similar results using $\Gamma = \Xi = 100$, and for each α_i, a logistic sigmoid function centered at the mean g differential of candidate translation pairs for the i^{th} source sentence. This alternative approach has the advantage of being agnostic about which gold scoring function is used.

[7] With the sampling settings previously described and MegaM as our classifier we were able to optimize two to three times faster than with MERT's line optimization.

PBMT					SBMT				
Language	Experiment		BLEU		Language	Experiment		BLEU	
	feats	method	tune	test		feats	method	tune	test
Urdu-English	base	MERT	20.5	17.7	Urdu-English	base	MERT	23.4	21.4
		MIRA	20.5	17.9			MIRA	23.6	22.3
		PRO	20.4	18.2			PRO	23.4	22.2
	ext	MIRA	21.8	17.8		ext	MIRA	25.2	22.8
		PRO	21.6	18.1			PRO	24.2	22.8
Arabic-English	base	MERT	46.8	41.2	Arabic-English	base	MERT	44.7	39.0
		MIRA	47.0	41.1			MIRA	44.6	39.0
		PRO	46.9	41.1			PRO	44.5	39.0
	ext	MIRA	47.5	41.7		ext	MIRA	45.8	39.8
		PRO	48.5	41.9			PRO	45.9	40.3
Chinese-English	base	MERT	23.8	22.2	Chinese-English	base	MERT	25.5	22.7
		MIRA	24.1	22.5			MIRA	25.4	22.9
		PRO	23.8	22.5			PRO	25.5	22.9
	ext	MIRA	24.8	22.6		ext	MIRA	26.0	23.3
		PRO	24.9	22.7			PRO	25.6	23.5

Table 1: Machine translation performance for the experiments listed in this paper. Scores are case-sensitive IBM BLEU. For every choice of system, language pair, and feature set, PRO performs comparably with the other methods.

5 Experiments

We now turn to real machine translation conditions to validate our thesis: We can cleanly replace MERT's line optimization with pairwise ranking optimization and immediately realize the benefits of high-dimension tuning. We now detail the three language pairs, two feature scenarios, and two MT models used for our experiments. For each language pair and each MT model we used MERT, MIRA, and PRO to tune with a standard set of baseline features, and used the latter two methods to tune with an extended set of features.[8] At the end of every experiment we used the final feature weights to decode a held-out test set and evaluated it with case-sensitive BLEU. The results are in Table 1.

5.1 Systems

We used two systems, each based on a different MT model. Our syntax-based system (hereafter, SBMT) follows the model of Galley et al. (2004). Our phrase-based system (hereafter, PBMT) follows the model of Och and Ney (2004). In both systems we learn alignments with GIZA++ (Och and Ney, 2000) using IBM Model 4; for Urdu-English and Chinese-English we merged alignments with the refined method, and for Arabic-English we merged with the union method.

5.2 Data

Table 2 notes the sizes of the datasets used in our experiments. All tune and test data have four English reference sets for the purposes of scoring.

Data		U-E	A-E	C-E
Train	lines	515K	6.5M	7.9M
	words	2.2M	175M	173M
Tune	lines	923	1994	1615
	words	16K	65K	42K
Test	lines	938	1357	1357
	words	18K	47K	37K

Table 2: Data sizes for the experiments reported in this paper (English words shown).

[8] MERT could not run to a satisfactory completion in any extended feature scenario; as implied in the synthetic data experiment of Section 3, the algorithm makes poor choices for its weights and this leads to low-quality k-best lists and dismal performance, near 0 BLEU in every iteration.

Class	Urdu-English				Arabic-English				Chinese-English			
	PBMT		SBMT		PBMT		SBMT		PBMT		SBMT	
	base	ext	base	ext	base	ext	base	ext	base	ext	base	ext
baseline	15	15	19	19	15	15	19	19	15	15	19	19
target word	–	51	–	50	–	51	–	50	–	51	–	299
discount	–	11	–	11	–	11	–	10	–	11	–	10
node count	–	–	–	99	–	–	–	138	–	–	–	96
rule overlap	–	–	–	98	–	–	–	136	–	–	–	93
word pair	–	2110	–	–	–	6193	–	–	–	1688	–	–
phrase length	–	63	–	–	–	63	–	–	–	63	–	–
total	15	2250	19	277	15	6333	18	352	15	1828	19	517

Table 3: Summary of features used in experiments in this paper.

5.2.1 Urdu-English

The training data for Urdu-English is that made available in the constrained track in the NIST 2009 MT evaluation. This includes many lexicon entries and other single-word data, which accounts for the large number of lines relative to word count. The NIST 2008 evaluation set, which contains newswire and web data, is split into two parts; we used roughly half each for tune and test. We trained a 5-gram English language model on the English side of the training data.

5.2.2 Arabic-English

The training data for Arabic English is that made available in the constrained track in the NIST 2008 MT evaluation. The tune set, which contains only newswire data, is a mix from NIST MT evaluation sets from 2003–2006 and from GALE development data. The test set, which contains both web and newswire data, is the evaluation set from the NIST 2008 MT evaluation. We trained a 4-gram English language model on the English side of the training data.

5.2.3 Chinese-English

For Chinese-English we used 173M words of training data from GALE 2008. For SBMT we used a 32M word subset for extracting rules and building a language model, but used the entire training data for alignments, and for all PBMT training. The tune and test sets both contain web and newswire data. The tune set is selected from NIST MT evaluation sets from 2003–2006. The test set is the evaluation set from the NIST 2008 MT evaluation. We trained a 3-gram English language model on the English side of the training data.

5.3 Features

For each of our systems we identify two feature sets: *baseline*, which correspond to the typical small feature set reported in current MT literature, and *extended*, a superset of baseline, which adds hundreds or thousands of features. Specifically, we use 15 baseline features for PBMT, similar to the baseline features described by Watanabe et al. (2007). We use 19 baseline features for SBMT, similar to the baseline features described by Chiang et al. (2008b).

We used the following feature classes in SBMT and PBMT extended scenarios:

- Discount features for rule frequency bins (cf. Chiang et al. (2009), Section 4.1)
- Target word insertion features[9]

We used the following feature classes in SBMT extended scenarios only (cf. Chiang et al. (2009), Section 4.1):[10]

- Rule overlap features
- Node count features

[9]For Chinese-English and Urdu-English SBMT these features only fired when the inserted target word was unaligned to any source word.

[10]The parser used for Arabic-English had a different nonterminal set than that used for the other two SBMT systems, accounting for the wide disparity in feature count for these feature classes.

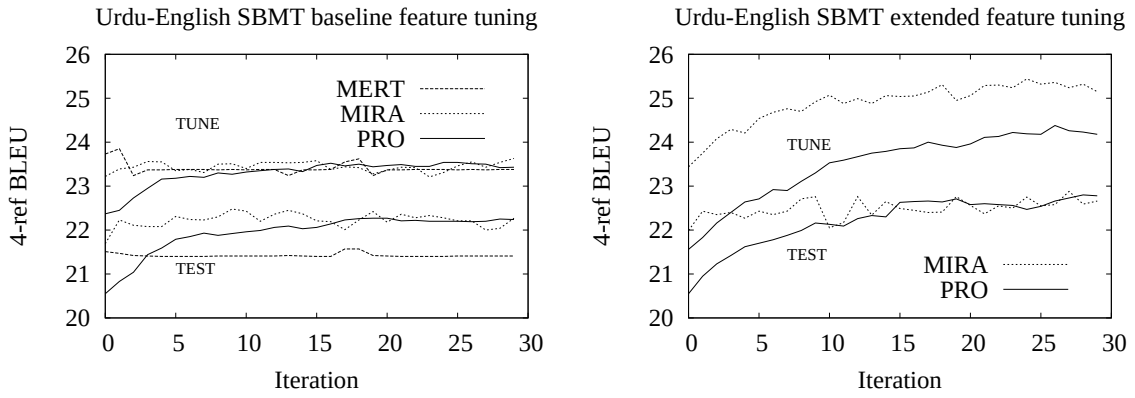


Figure 5: Comparison of MERT, PRO, and MIRA on tuning Urdu-English SBMT systems, and test results at every iteration. PRO performs comparably to MERT and MIRA.

We used the following feature classes in PBMT extended scenarios only:

- Unigram word pair features for the 80 most frequent words in both languages plus tokens for unaligned and all other words (cf. Watanabe et al. (2007), Section 3.2.1)[11]
- Source, target, and joint phrase length features from 1 to 7, e.g. "tgt=4", "src=2", and "src/tgt=2,4"

The feature classes and number of features used within those classes for each language pair are summarized in Table 3.

5.4 Tuning settings

Each of the three approaches we compare in this study has various details associated with it that may prove useful to those wishing to reproduce our results. We list choices made for the various tuning methods here, and note that all our decisions were made in keeping with best practices for each algorithm.

5.4.1 MERT

We used David Chiang's CMERT implementation of MERT that is available with the Moses system (Koehn et al., 2007). We ran MERT for up to 30 iterations, using $k = 1500$, and stopping early when the accumulated k-best list does not change in an iteration. In every tuning iteration we ran MERT once with weights initialized to the last iteration's chosen weight set and 19 times with random weights, and chose the the best of the 20 ending points according to G on the development set. The G we optimize is tokenized, lower-cased 4-gram BLEU (Papineni et al., 2002).

5.4.2 MIRA

We for the most part follow the MIRA algorithm for machine translation as described by Chiang et al. (2009)[12] but instead of using the 10-best of each of the best $h_{\mathbf{w}}$, $h_{\mathbf{w}}$ +g, and $h_{\mathbf{w}}$-g, we use the 30-best according to $h_{\mathbf{w}}$.[13] We use the same sentence-level BLEU calculated in the context of previous 1-best translations as Chiang et al. (2008b; 2009). We ran MIRA for 30 iterations.

5.4.3 PRO

We used the MegaM classifier and sampled as described in Section 4.2. As previously noted, we used BLEU+1 (Liang et al., 2006) for g. MegaM was easy to set up and ran fairly quickly, however any linear binary classifier that operates on real-valued features can be used, and in fact we obtained similar results

[11] This constitutes 6,723 features in principle ($82^2 - 1$ since "unaligned-unaligned" is not considered) but in practice far fewer co-occurrences were seen. Table 3 shows the number of actual unigram word pair features observed in data.

[12] and acknowledge the use of David Chiang's code

[13] This is a more realistic scenario for would-be implementers of MIRA, as obtaining the so-called "hope" and "fear" translations from the lattice or forest is significantly more complicated than simply obtaining a k-best list. Other tests comparing these methods have shown between 0.1 to 0.3 BLEU drop using 30-best $h_{\mathbf{w}}$ on Chinese-English (Wang, 2011).

using the support vector machine module of WEKA (Hall et al., 2009) as well as the Stanford classifier (Manning and Klein, 2003). We ran for up to 30 iterations and used the same k and stopping criterion as was used for MERT, though variability of sampling precluded list convergence.

While MERT and MIRA use each iteration's final weights as a starting point for hill-climbing the next iteration, the pairwise ranking approach has no explicit tie to previous iterations. To incorporate such stability into our process we interpolated the weights $\mathbf{w}'$ learned by the classifier in iteration t with those from iteration $t-1$ by a factor of Ψ, such that $\mathbf{w}_t = \Psi \cdot \mathbf{w}' + (1 - \Psi) \cdot \mathbf{w}_{t-1}$. We found $\Psi = 0.1$ gave good performance across the board.

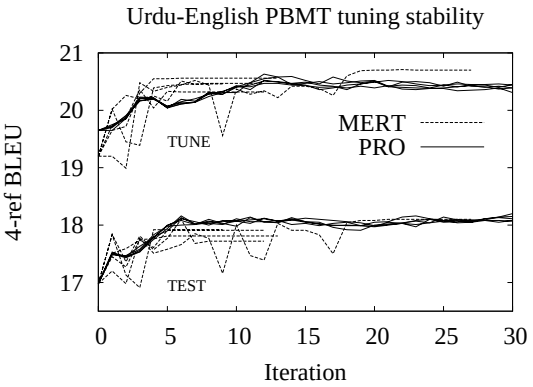


Figure 6: Tune and test curves of five repetitions of the same Urdu-English PBMT baseline feature experiment. PRO is more stable than MERT.

5.5 Discussion

We implore the reader to avoid the natural tendency to compare results using baseline vs. extended features or between PBMT and SBMT on the same language pair. Such discussions are indeed interesting, and could lead to improvements in feature engineering or sartorial choices due to the outcome of wagers (Goodale, 2008), but they distract from our thesis. As can be seen in Table 1, for each of the 12 choices of system, language pair, and feature set, the PRO method performed nearly the same as or better than MIRA and MERT on test data.

In Figure 5 we show the tune and test BLEU using the weights learned at every iteration for each Urdu-English SBMT experiment. Typical of the rest of the experiments, we can clearly see that PRO appears to proceed more monotonically than the other methods. We quantified PRO's stability as compared to MERT by repeating the Urdu-English baseline PBMT experiment five times with each configuration. The tune and test BLEU at each iteration is depicted in Figure 6. The standard deviation of the final test BLEU of MERT was 0.13 across the five experiment instances, while PRO had a standard deviation of just 0.05.

6 Related Work

Several works (Shen et al., 2004; Cowan et al., 2006; Watanabe et al., 2006) have used discriminative techniques to re-rank k-best lists for MT. Tillmann and Zhang (2005) used a customized form of multi-class stochastic gradient descent to learn feature weights for an MT model. Och and Ney (2002) used maximum entropy to tune feature weights but did not compare pairs of derivations. Ittycheriah and Roukos (2005) used a maximum entropy classifier to train an alignment model using hand-labeled data. Xiong et al. (2006) also used a maximum entropy classifier, in this case to train the reordering component of their MT model. Lattice- and hypergraph-based variants of MERT (Macherey et al., 2008; Kumar et al., 2009) are more stable than traditional MERT, but also require significant engineering efforts.

7 Conclusion

We have described a simple technique for tuning an MT system that is on par with the leading techniques, exhibits reliable behavior, scales gracefully to high-dimension feature spaces, and is remarkably easy to implement. We have demonstrated, via a litany of experiments, that our claims are valid and that this technique is widely applicable. It is our hope that the adoption of PRO tuning leads to fewer headaches during tuning and motivates advanced MT feature engineering research.

Acknowledgments

Thanks to Markus Dreyer, Kevin Knight, Saiyam Kohli, Greg Langmead, Daniel Marcu, Dragos Munteanu, and Wei Wang for their assistance. Thanks also to the anonymous reviewers, especially the reviewer who implemented PRO during the review period and replicated our results.

References

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 89–96, Bonn, Germany. ACM.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, Corvalis, OR.

Samidh Chatterjee and Nicola Cancedda. 2010. Minimum error rate training by sampling the translation lattice. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Cambridge, MA, October. Association for Computational Linguistics.

Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. 2009. Ranking measures and loss functions in learning to rank. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 315–323.

David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. 2008a. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 610–619, Honolulu, HI, October. Association for Computational Linguistics.

David Chiang, Yuval Marton, and Philip Resnik. 2008b. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Honolulu, HI, October. Association for Computational Linguistics.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Boulder, CO, June. Association for Computational Linguistics.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July. Association for Computational Linguistics.

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 232–241, Sydney, Australia, July. Association for Computational Linguistics.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`, August.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, MA, May. Association for Computational Linguistics.

Gloria Goodale. 2008. Language Weaver: fast in translation. *The Christian Science Monitor*, October 1. `http://www.csmonitor.com/Innovation/Tech-Culture/2008/1001/language-weaver-fast-in-translation`.

Jan Hajič, Sandra Carberry, Stephen Clark, and Joakim Nivre, editors. 2010. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, July.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1).

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression. In *Proceedings of the 1999 International Conference on Artificial Neural Networks*, pages 97–102.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 89–96, Vancouver, Canada, October. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–

180, Prague, Czech Republic, June. Association for Computational Linguistics.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171, Suntec, Singapore, August. Association for Computational Linguistics.

Alon Lavie and Michael J. Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2–3):105–115, September.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July. Association for Computational Linguistics.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734, Honolulu, HI, October. Association for Computational Linguistics.

Christopher Manning and Dan Klein. 2003. Optimization, maxent models, and conditional estimation without magic. Tutorial at HLT-NAACL 2003 and ACL 2003.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. 2003. Precision and recall of machine translation. In *Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers*, pages 61–63, Edmonton, Canada, May–June. Association for Computational Linguistics.

Franz Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong, October.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, PA, July. Association for Computational Linguistics.

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

Franz Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, July. Association for Computational Linguistics.

Benjamin Roth, Andrew McCallum, Marc Dymetman, and Nicola Cancedda. 2010. Machine translation using overlapping alignments and samplerank. In *Proceedings of Association for Machine Translation in the Americas*, Denver, CO.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 177–184, Boston, MA, May 2 - May 7. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Christoph Tillmann and Tong Zhang. 2005. A localized prediction model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 557–564, Ann Arbor, MI, June. Association for Computational Linguistics.

Wei Wang. 2011. Personal communication.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2006. NTT statistical machine translation for IWSLT 2006. In *Proceedings of IWSLT 2006*, pages 95–102.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic, June. Association for Computational Linguistics.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528, Sydney, Australia, July. Association for Computational Linguistics.

Watermarking the Outputs of Structured Prediction with an application in Statistical Machine Translation.

Ashish Venugopal[1] **Jakob Uszkoreit**[1] **David Talbot**[1] **Franz J. Och**[1] **Juri Ganitkevitch**[2]

[1]Google, Inc.
1600 Amphitheatre Parkway
Mountain View, 94303, CA
{avenugopal,uszkoreit,talbot,och}@google.com

[2]Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218, USA
juri@cs.jhu.edu

Abstract

We propose a general method to watermark and probabilistically identify the structured outputs of machine learning algorithms. Our method is robust to local editing operations and provides well defined trade-offs between the ability to identify algorithm outputs and the quality of the watermarked output. Unlike previous work in the field, our approach does not rely on controlling the inputs to the algorithm and provides probabilistic guarantees on the ability to identify collections of results from one's own algorithm. We present an application in statistical machine translation, where machine translated output is watermarked at minimal loss in translation quality and detected with high recall.

1 Motivation

Machine learning algorithms provide structured results to input queries by simulating human behavior. Examples include automatic machine translation (Brown et al., 1993) or automatic text and rich media summarization (Goldstein et al., 1999). These algorithms often estimate some portion of their models from publicly available human generated data. As new services that output structured results are made available to the public and the results disseminated on the web, we face a daunting new challenge: Machine generated structured results contaminate the pool of naturally generated human data. For example, machine translated output and human generated translations are currently both found extensively on the web, with no automatic way of distinguishing between them. Algorithms that mine data from the web (Uszkoreit et al., 2010), with the goal of learning to simulate human behavior, will now learn models from this contaminated and potentially self-generated data, reinforcing the errors committed by earlier versions of the algorithm.

It is beneficial to be able to identify a set of encountered structured results as having been generated by one's own algorithm, with the purpose of filtering such results when building new models.

Problem Statement: We define a structured result of a query q as $r = \{z_1 \cdots z_L\}$ where the order and identity of elements z_i are important to the quality of the result r. The structural aspect of the result implies the existence of alternative results (across both the order of elements and the elements themselves) that might vary in their quality.

Given a collection of N results, $\mathcal{C}_N = r_1 \cdots r_N$, where each result r_i has k ranked alternatives $D_k(q_i)$ of relatively similar quality and queries $q_1 \cdots q_N$ are arbitrary and not controlled by the watermarking algorithm, we define the watermarking task as:

Task. *Replace r_i with $r'_i \in D_k(q_i)$ for some subset of results in $\mathcal{C}_N$ to produce a watermarked collection $\mathcal{C}'_N$*

such that:

- $\mathcal{C}'_N$ is probabilistically identifiable as having been generated by one's own algorithm.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1363–1372,
Edinburgh, Scotland, UK, July 27–31, 2011.

- the degradation in quality from $\mathcal{C}_N$ to the watermarked $\mathcal{C}'_N$ should be analytically controllable, trading quality for detection performance.
- $\mathcal{C}'_N$ should not be detectable as watermarked content without access to the generating algorithms.
- the detection of $\mathcal{C}'_N$ should be robust to simple edit operations performed on individual results $r \in \mathcal{C}'_N$.

2 Impact on Statistical Machine Translation

Recent work(Resnik and Smith, 2003; Munteanu and Marcu, 2005; Uszkoreit et al., 2010) has shown that multilingual parallel documents can be efficiently identified on the web and used as training data to improve the quality of statistical machine translation.

The availability of free translation services (Google Translate, Bing Translate) and tools (Moses, Joshua), increase the risk that the content found by parallel data mining is in fact generated by a machine, rather than by humans. In this work, we focus on statistical machine translation as an application for watermarking, with the goal of discarding documents from training if they have been generated by one's own algorithms.

To estimate the magnitude of the problem, we used parallel document mining (Uszkoreit et al., 2010) to generate a collection of bilingual document pairs across several languages. For each document, we inspected the page content for source code that indicates the use of translation modules/plug-ins that translate and publish the translated content.

We computed the proportion of the content within our corpus that uses these modules. We find that a significant proportion of the mined parallel data for some language pairs is generated via one of these translation modules. The top 3 languages pairs, each with parallel translations into English, are Tagalog (50.6%), Hindi (44.5%) and Galician (41.9%). While these proportions do not reflect impact on each language's monolingual web, they are certainly high enough to affect machine translations systems that train on mined parallel data. In this work, we develop a general approach to watermark structured outputs and apply it to the outputs of a statistical machine translation system with the goal of identifying these same outputs on the web. In the context of the watermarking task defined above, we output selecting alternative translations for input source sentences. These translations often undergo simple edit and formatting operations such as case changes, sentence and word deletion or post editing, prior to publishing on the web. We want to ensure that we can still detect watermarked translations despite these edit operations. Given the rapid pace of development within machine translation, it is also important that the watermark be robust to improvements in underlying translation quality. Results from several iterations of the system within a single collection of documents should be identifiable under probabilistic bounds.

While we present evaluation results for statistical machine translation, our proposed approach and associated requirements are applicable to any algorithm that produces structured results with several plausible alternatives. The alternative results can arise as a result of inherent task ambiguity (for example, there are multiple correct translations for a given input source sentence) or modeling uncertainty (for example, a model assigning equal probability to two competing results).

3 Watermark Structured Results

Selecting an alternative r' from the space of alternatives $D_k(q)$ can be stated as:

$$r' = \arg\max_{r \in D_k(q)} w(r, D_k(q), h) \quad (1)$$

where w ranks $r \in D_k(q)$ based on r's presentation of a watermarking signal computed by a hashing operation h. In this approach, w and its component operation h are the only *secrets* held by the watermarker. This selection criterion is applied to all system outputs, ensuring that watermarked and non-watermarked version of a collection will never be available for comparison.

A specific implementation of w within our watermarking approach can be evaluated by the following metrics:

- False Positive Rate: how often non-watermarked collections are *falsely* identified as watermarked.
- Recall Rate: how often watermarked collections are *correctly* identified as watermarked.
- Quality Degradation: how significantly does $\mathcal{C}'_N$ differ from $\mathcal{C}_N$ when evaluated by task specific quality metrics.

While identification is performed at the collection level, we can scale these metrics based on the size of each collection to provide more task sensitive metrics. For example, in machine translation, we count the number of words in the collection towards the false positive and recall rates.

In Section 3.1, we define a random hashing operation h and a task independent implementation of the selector function w. Section 3.2 describes how to classify a collection of watermarked results. Section 3.3 and 3.4 describes refinements to the selection and classification criteria that mitigate quality degradation. Following a comparison to related work in Section 4, we present experimental results for several languages in Section 5.

3.1 Watermarking: $\mathcal{C}_N \rightarrow \mathcal{C}'_N$

We define a random hashing operation h that is applied to result r. It consists of two components:

- A hash function applied to a structured result r to generate a bit sequence of a fixed length.
- An optional mapping that maps a single candidate result r to a set of sub-results. Each sub-result is then hashed to generate a concatenated bit sequence for r.

A good hash function produces outputs whose bits are independent. This implies that we can treat the bits for any input structured results as having been generated by a binomial distribution with equal probability of generating 1s vs 0s. This condition also holds when accumulating the bit sequences over a collection of results as long as its elements are selected uniformly from the space of possible results. Therefore, *the bits generated from a collection of unwatermarked results will follow a binomial distribution with parameter* $p = 0.5$. This result provides a null hypothesis for a statistical test on a given bit sequence, testing whether it is likely to have been generated from a binomial distribution binomial(n, p) where $p = 0.5$ and n is the length of the bit sequence.

For a collection $\mathcal{C}_N = r_1 \cdots r_N$, we can define a watermark ranking function w to systematically select alternatives $r'_i \in D_k(q)$, such that the resulting $\mathcal{C}'_N$ is *unlikely* to produce bit sequences that follow the $p = 0.5$ binomial distribution. A straightforward biasing criteria would be to select the candidate whose bit sequence exhibits the highest ratio of 1s. w can be defined as:

$$w(r, D_k(q), h) = \frac{\#(1, h(r))}{|h(r)|} \qquad (2)$$

where $h(r)$ returns the randomized bit sequence for result r, and $\#(x, \vec{y})$ counts the number of occurrences of x in sequence $\vec{y}$. Selecting alternatives results to exhibit this bias will result in watermarked collections that exhibit this same bias.

3.2 Detecting the Watermark

To classify a collection $\mathcal{C}_N$ as watermarked or non-watermarked, we apply the hashing operation h on each element in $\mathcal{C}_N$ and concatenate the sequences. This sequence is tested against the null hypothesis that it was generated by a binomial distribution with parameter $p = 0.5$. We can apply a Fisherian test of statistical significance to determine whether the observed distribution of bits is unlikely to have occurred by chance under the null hypothesis (binomial with $p = 0.5$).

We consider a collection of results that *rejects* the null hypothesis to be watermarked results generated by our own algorithms. The p-value under the null hypothesis is efficiently computed

by:

$$p - value = P_n(X \geq x) \quad (3)$$

$$= \sum_{i=x}^{n} \binom{n}{i} p^i (1-p)^{n-i} \quad (4)$$

where x is the number of 1s observed in the collection, and n is the total number of bits in the sequence. Comparing this p-value against a desired significance level α, we reject the null hypothesis for collections that have $P_n(X \geq x) < \alpha$, thus deciding that such collections were generated by our own system.

This classification criteria has a fixed false positive rate. Setting $\alpha = 0.05$, we know that 5% of *non-watermarked* bit sequences will be *falsely* labeled as watermarked. This parameter α can be controlled on an application specific basis. By biasing the selection of candidate results to produce more 1s than 0s, we have defined a watermarking approach that exhibits a fixed false positive rate, a probabilistically bounded detection rate and a task independent hashing and selection criteria. In the next sections, we will deal with the question of robustness to edit operations and quality degradation.

3.3 Robustness and Inherent Bias

We would like the ability to identify watermarked collections to be robust to simple edit operations. Even slight modifications to the elements within an item r would yield (by construction of the hash function), completely different bit sequences that no longer preserve the biases introduced by the watermark selection function.

To ensure that the distributional biases introduced by the watermark selector are preserved, we can optionally map individual results into a set of sub-results, each one representing some local structure of r. h is then applied to each sub-result and the results concatenated to represent r. This mapping is defined as a component of the h operation.

While a particular edit operation might affect a small number of sub-results, the majority of the bits in the concatenated bit sequence for r would remain untouched, thereby limiting the damage to the biases selected during watermarking. This is of course no defense to edit operations that are applied globally across the result; our expectation is that such edits would either significantly degrade the quality of the result or be straightforward to identify directly.

For example, a sequence of words $r = z_1 \cdots z_L$ can be mapped into a set of consecutive n-gram sequences. Operations to edit a word z_i in r will only affect events that consider the word z_i. To account for the fact that alternatives in $D_k(q)$ might now result in bit sequences of different lengths, we can generalize the biasing criteria to directly reflect the expected contribution to the watermark by defining:

$$w(r, D_k(q), h) = P_n(X \geq \#(1, h(r))) \quad (5)$$

where P_n gives probabilities from binomial($n = |h(r)|, p = 0.5$).

Inherent collection level biases: Our null hypothesis is based on the assumption that collections of results draw uniformly from the space of possible results. This assumption might not always hold and depends on the type of the results and collection. For example, considering a text document as a collection of sentences, we can expect that some sentences might repeat more frequently than others.

This scenario is even more likely when applying a mapping into sub-results. n-gram sequences follow long-tailed or Zipfian distributions, with a small number of n-grams contributing heavily toward the total number of n-grams in a document.

A random hash function guarantees that inputs are distributed uniformly at random over the output range. However, the same input will be assigned the same output deterministically. Therefore, if the distribution of inputs is heavily skewed to certain elements of the input space, the output distribution will not be uniformly distributed. The bit sequences resulting from the high frequency sub-results have the potential to generate inherently biased distributions when accumulated at the collection level. We want to choose a mapping that tends towards generating uniformly from the space of sub-results. We can empirically measure the quality of a sub-result mapping for a specific task by computing the

false positive rate on non-watermarked collections. For a given significance level α, an ideal mapping would result in false positive rates close to α as well.

Figure 1 shows false positive rates from 4 alternative mappings, computed on a large corpus of French documents (see Table 1 for statistics). Classification decisions are made at the collection level (documents) but the contribution to the false positive rate is based on the number of words in the classified document. We consider mappings from a result (sentence) into its 1-grams, $1-5$-grams and $3-5$ grams as well as the non-mapping case, where the full result is hashed.

Figure 1 shows that the 1-grams and $1-5$-gram generate sub-results that result in heavily biased false positive rates. The $3-5$ gram mapping yields false positive rates close to their theoretically expected values. [1] Small deviations are expected since documents make different contributions to the false positive rate as a function of the number of words that they represent. For the remainder of this work, we use the 3-5 gram mapping and the full sentence mapping, since the alternatives generate inherently distributions with very high false positive rates.

3.4 Considering Quality

The watermarking described in Equation 3 chooses alternative results on a per result basis, with the goal of influencing collection level bit sequences. The selection criteria as described will choose the most biased candidates available in $D_k(q)$. The parameter k determines the extent to which lesser quality alternatives can be chosen. If all the alternatives in each $D_k(q)$ are of relatively similar quality, we expect minimal degradation due to watermarking.

Specific tasks however can be particularly sensitive to choosing alternative results. Discriminative approaches that optimize for arg max selection like (Och, 2003; Liang et al., 2006; Chiang et al., 2009) train model parameters such that the top-ranked result is well *separated* from its competing alternatives. Different queries also differ in the inherent ambiguity expected from their results; sometimes there really is just one correct result for a query, while for other queries, several alternatives might be equally good.

By generalizing the definition of the w function to interpolate the estimated loss in quality and the gain in the watermarking signal, we can trade-off the ability to identify the watermarked collections against quality degradation:

$$\begin{aligned} w(r, D_k(q), f_w) = \lambda * \text{gain}(r, D_k(q), f_w) \\ -(1-\lambda) * \text{loss}(r, D_k(q)) \end{aligned} \quad (6)$$

Loss: The $\text{loss}(r, D_k(q))$ function reflects the quality degradation that results from selecting alternative r as opposed to the best ranked candidate in $D_k(q)$). We will experiment with two variants:

$$loss_{rank}(r, D_k(q)) = (\text{rank}(r) - k)/k$$

$$loss_{cost}(r, D_k(q)) = (\text{cost}(r) - \text{cost}(r^1))/\,\text{cost}(r^1)$$

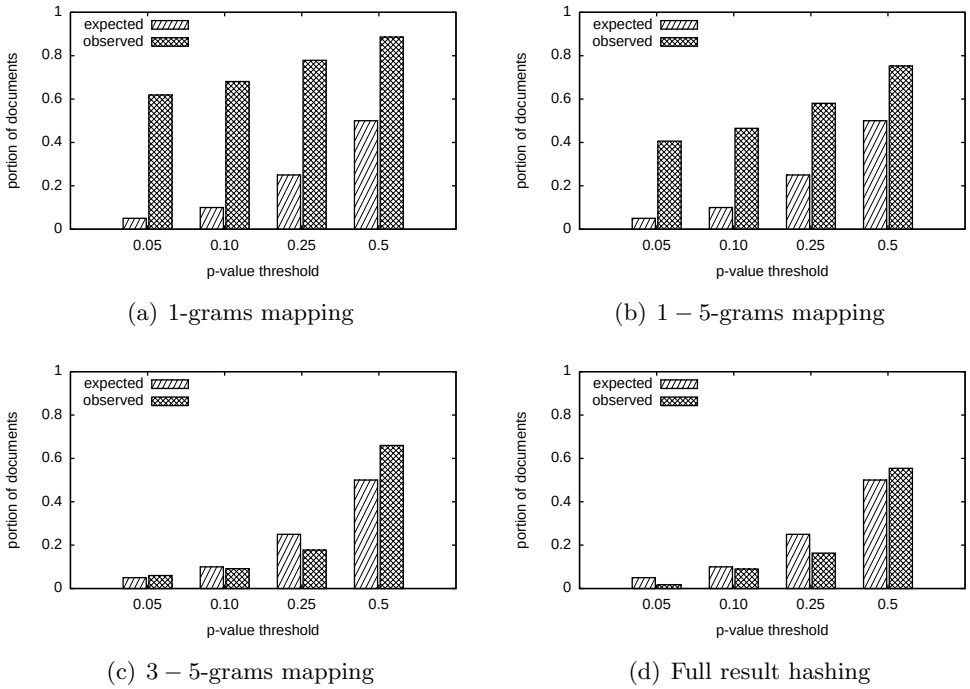
where:

- $\text{rank}(r)$: returns the rank of r within $D_k(q)$.
- $\text{cost}(r)$: a weighted sum of features (not normalized over the search space) in a log-linear model such as those mentioned in (Och, 2003).
- r^1: the highest ranked alternative in $D_k(q)$.

loss_{rank} provides a generally applicable criteria to select alternatives, penalizing selection from deep within $D_k(q)$. This estimate of the quality degradation does not reflect the generating model's opinion on relative quality. loss_{cost} considers the relative increase in the generating model's cost assigned to the alternative translation.

Gain: The $\text{gain}(r, D_k(q), f_w)$ function reflects the gain in the watermarking signal by selecting candidate r. We simply define the gain as the $P_n(X \geq \#(1, h(r)))$ from Equation 5.

[1] In the final version of this paper we will perform sampling to create a more reliable estimate of the false positive rate that is not overly influenced by document length distributions.



(a) 1-grams mapping

(b) $1-5$-grams mapping

(c) $3-5$-grams mapping

(d) Full result hashing

Figure 1: Comparison of expected false positive rates against observed false positive rates for different sub-result mappings.

4 Related Work

Using watermarks with the goal of transmitting a hidden message within images, video, audio and monolingual text media is common. For structured text content, linguistic approaches like (Chapman et al., 2001; Gupta et al., 2006) use language specific linguistic and semantic expansions to introduce hidden watermarks. These expansions provide alternative candidates within which messages can be encoded. Recent publications have extended this idea to machine translation, using multiple systems and expansions to generate alternative translations. (Stutsman et al., 2006) uses a hashing function to select alternatives that encode the hidden message in the lower order bits of the translation. In each of these approaches, the watermarker has control over the collection of results into which the watermark is to be embedded.

These approaches seek to embed a hidden message into a collection of results that is *selected* by the watermarker. In contrast, we address the condition where the input queries are not in the watermarker's control.

The goal is therefore to introduce the watermark into all generated results, with the goal of probabilistically identifying such outputs. Our approach is also task independent, avoiding the need for templates to generate additional alternatives. By addressing the problem directly within the search space of a dynamic programming algorithm, we have access to high quality alternatives with well defined models of quality loss. Finally, our approach is robust to local word editing. By using a sub-result mapping, we increase the level of editing required to obscure the watermark signal; at high levels of editing, the quality of the results themselves would be significantly degraded.

5 Experiments

We evaluate our watermarking approach applied to the outputs of statistical machine translation under the following experimental setup.

A repository of parallel (aligned source and target language) web documents is sampled to produce a large corpus on which to evaluate the watermarking *classification* performance. The

corpora represent translations into 4 diverse target languages, using English as the source language. Each document in this corpus can be considered a collection of un-watermarked structured results, where source sentences are queries and each target sentence represents a structured result.

Using a state-of-the-art phrase-based statistical machine translation system (Och and Ney, 2004) trained on parallel documents identified by (Uszkoreit et al., 2010), we generate a set of 100 alternative translations for each source sentence. We apply the proposed watermarking approach, along with the proposed refinements that address task specific loss (Section 3.4) and robustness to edit operations (Section 3.3) to generate watermarked corpora.

Each method is controlled via a single parameter (like k or λ) which is varied to generate alternative watermarked collections. For each parameter value, we evaluate the Recall Rate and Quality Degradation with the goal of finding a setting that yields a high recall rate, minimal quality degradation. False positive rates are evaluated based on a fixed classification significance level of $\alpha = 0.05$. The false positive and recall rates are evaluated on the word level; a document that is misclassified or correctly identified contributes its length in words towards the error calculation. In this work, we use $\alpha = 0.05$ during classification corresponding to an expected 5% false positive rate. The false positive rate is a function of h and the significance level α and therefore constant across the parameter values k and λ.

We evaluate quality degradation on human translated test corpora that are more typical for machine translation evaluation. Each test corpus consists of 5000 source sentences randomly selected from the web and translated into each respective language.

We chose to evaluate quality on test corpora to ensure that degradations are not hidden by imperfectly matched web corpora and are consistent with the kind of results often reported for machine translation systems. As with the classification corpora, we create watermarked versions at each parameter value. For a given parameter value, we measure false positive and recall rates on the classification corpora and quality degradation on the evaluation corpora.

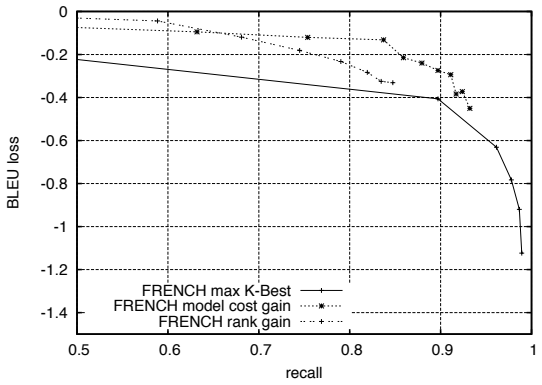


Figure 2: BLEU loss against recall of watermarked content for the baseline approach (max K-best), rank and cost interpolation.

Table 1 shows corpus statistics for the classification and test corpora and non-watermarked BLEU scores for each target language. All source texts are in English.

5.1 Loss Interpolated Experiments

Our first set of experiments demonstrates baseline performance using the watermarking criteria in Equation 5 versus the refinements suggested in Section 3.4 to mitigate quality degradation. The h function is computed on the full sentence result r with no sub-event mapping. The following methods are evaluated in Figure 2.

- Baseline method (labeled "max K-best"): selects r' purely based on gain in watermarking signal (Equation 5) and is parameterized by k: the number of alternatives considered for each result.
- Rank interpolation: incorporates rank into w, varying the interpolation parameter λ.
- Cost interpolation: incorporates cost into w, varying the interpolation parameter λ.

The observed false positive rate on the French classification corpora is 1.9%.

	Classification			Quality		
Target	**# words**	**# sentences**	**# documents**	**# words**	**# sentences**	**BLEU %**
Arabic	200107	15820	896	73592	5503	12.29
French	209540	18024	600	73592	5503	26.45
Hindi	183676	13244	1300	73409	5489	20.57
Turkish	171671	17155	1697	73347	5486	13.67

Table 1: Content statistics for classification and quality degradation corpora. Non-watermarked BLEU scores are reported for the quality corpora.

We consider 0.2% BLEU loss as a threshold for acceptable quality degradation. Each method is judged by its ability to achieve high recall below this quality degradation threshold.

Applying cost interpolation yields the best results in Figure 2, achieving a recall of 85% at 0.2% BLEU loss, while rank interpolation achieves a recall of 76%. The baseline approach of selecting the highest gain candidate within a depth of k candidates does not provide sufficient parameterization to yield low quality degradation. At $k = 2$, this method yields almost 90% recall, but with approximately 0.4% BLEU loss.

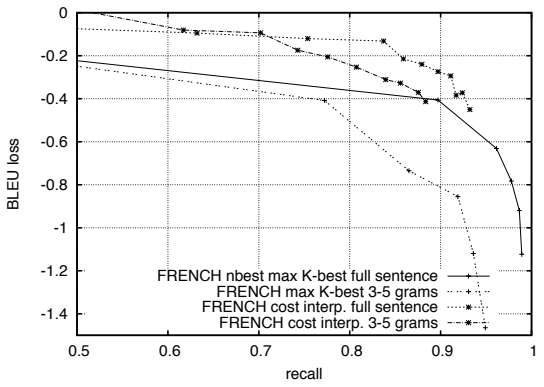


Figure 3: BLEU loss against recall of watermarked content for the baseline and cost interpolation methods using both result level and 3-5 gram mapped events.

5.2 Robustness Experiments

In Section 5.2, we proposed mapping results into sub-events or features. We considered alternative feature mappings in Figure 1, finding that mapping sentence results into a collection of 3-5 grams yields acceptable false positive rates at varied levels of α.

Figure 3 presents results that compare moving from the result level hashing to the 3-5 gram sub-result mapping. We show the impact of the mapping on the baseline max K-best method as well as for cost interpolation. There are substantial reductions in recall rate at the 0.2% BLEU loss level when applying sub-result mappings in cases. The cost interpolation method recall drops from 85% to 77% when using the 3-5 grams event mapping. The observed false positive rate of the 3-5 gram mapping is 4.7%.

By using the 3-5 gram mapping, we expect to increase robustness against local word edit operations, but we have sacrificed recall rate due to the inherent distributional bias discussed in Section 3.3.

5.3 Multilingual Experiments

The watermarking approach proposed here introduces no language specific watermarking operations and it is thus broadly applicable to translating into all languages. In Figure 4, we report results for the baseline and cost interpolation methods, considering both the result level and 3-5 gram mapping. We set $\alpha = 0.05$ and measure recall at 0.2% BLEU degradation for translation from English into Arabic, French, Hindi and Turkish. The observed false positive rates for full sentence hashing are: Arabic: 2.4%, French: 1.8%, Hindi: 5.6% and Turkish: 5.5%, while for the 3-5 gram mapping, they are: Arabic: 5.8%, French: 7.5%, Hindi:3.5% and Turkish: 6.2%. Underlying translation quality plays an important role in translation quality degradation when watermarking. Without a sub-result mapping, French (BLEU: 26.45%)

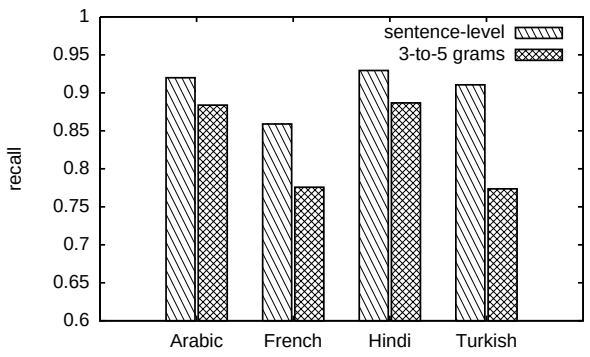


Figure 4: Loss of recall when using 3-5 gram mapping vs sentence level mapping for Arabic, French, Hindi and Turkish translations.

achieves recall of 85% at 0.2% BLEU loss, while the other languages achieve over 90% recall at the same BLEU loss threshold. Using a sub-result mapping degrades quality for each language pair, but changes the relative performance. Turkish experiences the highest relative drop in recall, unlike French and Arabic, where results are relatively more robust to using sub-sentence mappings. This is likely a result of differences in n-gram distributions across these languages. The languages considered here all use space separated words. For languages that do not, like Chinese or Thai, our approach can be applied at the character level.

6 Conclusions

In this work we proposed a general method to watermark and probabilistically identify the structured outputs of machine learning algorithms. Our method provides probabilistic bounds on detection ability, analytic control on quality degradation and is robust to local editing operations. Our method is applicable to any task where structured outputs are generated with ambiguities or ties in the results. We applied this method to the outputs of statistical machine translation, evaluating each refinement to our approach with false positive and recall rates against BLEU score quality degradation.

Our results show that it is possible, across several language pairs, to achieve high recall rates (over 80%) with low false positive rates (between 5 and 8%) at minimal quality degradation (0.2% BLEU), while still allowing for local edit operations on the translated output. In future work we will continue to investigate methods to mitigate quality loss.

References

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Minimum error rate training in statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

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

Mark Chapman, George Davida, and Marc Rennhardway. 2001. A practical and effective approach to large-scale automated linguistic steganography. In *Proceedings of the Information Security Conference*.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *Research and Development in Information Retrieval*, pages 121–128.

Gaurav Gupta, Josef Pieprzyk, and Hua Xiong Wang. 2006. An attack-localizing watermarking scheme for natural language documents. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ASIACCS '06, pages 157–165, New York, NY, USA. ACM.

Percy Liang, Alexandre Bouchard-Cote, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the Joint International Conference on Computational Linguistics and Association of Computational Linguistics (COLING/ACL*, pages 761–768.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*.

Franz Josef Och and Hermann Ney. 2004. The

alignment template approach to statistical machine translation. *Computational Linguistics*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 2003 Meeting of the Asssociation of Computational Linguistics*.

Philip Resnik and Noah A. Smith. 2003. The web as a parallel corpus. computational linguistics. *Computational Linguistics*.

Ryan Stutsman, Mikhail Atallah, Christian Grothoff, and Krista Grothoff. 2006. Lost in just the translation. In *Proceedings of the 2006 ACM Symposium on Applied Computing*.

Jakob Uszkoreit, Jay Ponte, Ashok Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proceedings of the 2010 COLING*.

Hierarchical Phrase-Based Translation Representations

Gonzalo Iglesias[⋆] **Cyril Allauzen**[‡] **William Byrne**[⋆]
Adrià de Gispert[⋆] **Michael Riley**[‡]

[⋆]Department of Engineering, University of Cambridge, Cambridge, CB2 1PZ, U.K.
{gi212,wjb31,ad465}@eng.cam.ac.uk
[‡] Google Research, 76 Ninth Avenue, New York, NY 10011
{allauzen,riley}@google.com

Abstract

This paper compares several translation representations for a synchronous context-free grammar parse including CFGs/hypergraphs, finite-state automata (FSA), and pushdown automata (PDA). The representation choice is shown to determine the form and complexity of target LM intersection and shortest-path algorithms that follow. Intersection, shortest path, FSA expansion and RTN replacement algorithms are presented for PDAs. Chinese-to-English translation experiments using HiFST and HiPDT, FSA and PDA-based decoders, are presented using admissible (or exact) search, possible for HiFST with compact SCFG rulesets and HiPDT with compact LMs. For large rulesets with large LMs, we introduce a two-pass search strategy which we then analyze in terms of search errors and translation performance.

1 Introduction

Hierarchical phrase-based translation, using a *synchronous context-free translation grammar* (SCFG) together with an n-gram target language model (LM), is a popular approach in machine translation (Chiang, 2007). Given a SCFG G and an n-gram language model M, this paper focuses on how to *decode* with them, i.e. how to apply them to the source text to generate a target translation. Decoding has three basic steps, which we first describe in terms of the formal languages and relations involved, with data representations and algorithms to follow.

1. *Translating the source sentence s with G to give target translations:* $\mathcal{T} = \{s\} \circ \mathcal{G}$, a (weighted) context-free language resulting from the composition of a finite language and the algebraic relation $\mathcal{G}$ for SCFG G.
2. *Applying the language model to these target translations:* $\mathcal{L} = \mathcal{T} \cap \mathcal{M}$, a (weighted) context-free language resulting from the intersection of a context-free language and the regular language $\mathcal{M}$ for M.
3. *Searching for the translation and language model combination with the highest-probablity path:* $\hat{\mathcal{L}} = \mathrm{argmax}_{l \in \mathcal{L}} L$

Of course, decoding requires explicit data representations and algorithms for combining and searching them. In common to the approaches we will consider here, s is applied to G by using the CYK algorithm in Step 1 and M is represented by a finite automaton in Step 2. The choice of the representation of $\mathcal{T}$ in many ways determines the remaining decoder representations and algorithms needed. Since $\{s\}$ is a finite language and we assume throughout that G does not allow unbounded insertions, $\mathcal{T}$ and $\mathcal{L}$ are, in fact, regular languages. As such, $\mathcal{T}$ and $\mathcal{L}$ have finite automaton representations T_f and L_f. In this case, weighted finite-state intersection and single-source shortest path algorithms (using negative log probabilities) can be used to solve Steps 2 and 3 (Mohri, 2009). This is the approach taken in (Iglesias et al., 2009a; de Gispert et al., 2010). Instead $\mathcal{T}$ and $\mathcal{L}$ can be represented by *hypergraphs* T_h and L_h (or very similarly context-free rules, and-or trees, or deductive systems). In this case, hypergraph intersection with a finite automaton and hypergraph shortest path algorithms can be used to solve Steps 2 and 3 (Huang, 2008). This is the approach taken by Chiang (2007). In this paper, we will consider another representation for context-free languages $\mathcal{T}$ and $\mathcal{L}$ as well, *pushdown automata* (PDA) T_p and L_p, familiar from formal

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1373–1383,
Edinburgh, Scotland, UK, July 27–31, 2011.

language theory (Aho and Ullman, 1972). We will describe PDA intersection with a finite automaton and PDA shortest-path algorithms in Section 2 that can be used to solve Steps 2 and 3. It cannot be over-emphasized that the CFG, hypergraph and PDA representations of $\mathcal{T}$ are used for their compactness rather than for expressing non-regular languages.

As presented so far, the search performed in Step 3 is *admissible* (or *exact*) – the true shortest path is found. However, the search space in MT can be quite large. Many systems employ aggressive pruning during the shortest-path computation with little theoretical or empirical guarantees of correctness. Further, such pruning can greatly complicate any complexity analysis of the underlying representations and algorithms. In this paper, we will exclude any inadmissible pruning in the shortest-path algorithm itself. This allows us in Section 3 to compare the computational complexity of using these different representations. We show that the PDA representation is particularly suited for decoding with large SCFGs and compact LMs.

We present Chinese-English translation results under the FSA and PDA translation representations. We describe a two-pass translation strategy which we have developed to allow use of the PDA representation in large-scale translation. In the first pass, translation is done using a lattice-generating version of the shortest path algorithm. The full translation grammar is used but with a compact, entropy-pruned version (Stolcke, 1998) of the full language model. This first-step uses admissible pruning and lattice generation under the compact language model. In the second pass, the original, unpruned LM is simply applied to the lattices produced in the first pass. We find that entropy-pruning and first-pass translation can be done so as to introduce very few search errors in the overall process; we can identify search errors in this experiment by comparison to exact translation under the full translation grammar and language model using the FSA representation. We then investigate a translation grammar which is large enough that exact translation under the FSA representation is not possible. We find that translation is possible using the two-pass strategy with the PDA translation representation and that gains in BLEU score result from using the larger translation grammar.

1.1 Related Work

There is extensive prior work on computational efficiency and algorithmic complexity in hierarchical phrase-based translation. The challenge is to find algorithms that can be made to work with large translation grammars and large language models.

Following the original algorithms and analysis of Chiang (2007), Huang and Chiang (2007) developed the cube-growing algorithm, and more recently Huang and Mi (2010) developed an incremental decoding approach that exploits left-to-right nature of the language models.

Search errors in hierarchical translation, and in translation more generally, have not been as extensively studied; this is undoubtedly due to the difficulties inherent in finding exact translations for use in comparison. Using a relatively simple phrase-based translation grammar, Iglesias et al. (2009b) compared search via cube-pruning to an exact FST implementation (Kumar et al., 2006) and found that cube-pruning suffered significant search errors. For Hiero translation, an extensive comparison of search errors between the cube pruning and FSA implementation was presented by Iglesias et al. (2009a) and de Gispert et al. (2010). Relaxation techniques have also recently been shown to finding exact solutions in parsing (Koo et al., 2010) and in SMT with tree-to-string translation grammars and trigram language models (Rush and Collins, 2011), much smaller models compared to the work presented in this paper.

Although entropy-pruned language models have been used to produce real-time translation systems (Prasad et al., 2007), we believe our use of entropy-pruned language models in two-pass translation to be novel. This is an approach that is widely-used in automatic speech recognition (Ljolje et al., 1999) and we note that it relies on efficient representation of very large search spaces $\mathcal{T}$ for subsequent rescoring, as is possible with FSAs and PDAs.

2 Pushdown Automata

In this section, we formally define pushdown automata and give intersection, shortest-path and related algorithms that will be needed later.

Informally, pushdown automata are finite automata that have been augmented with a stack. Typ-

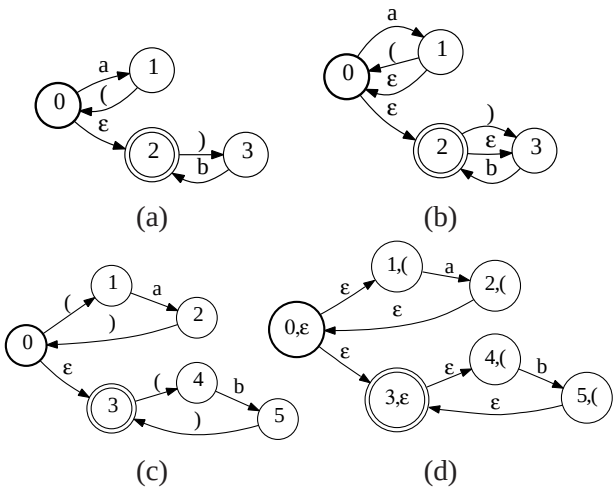


Figure 1: PDA Examples: (a) Non-regular PDA accepting $\{a^n b^n | n \in \mathbb{N}\}$. (b) Regular (but not bounded-stack) PDA accepting a^*b^*. (c) Bounded-stack PDA accepting a^*b^* and (d) its expansion as an FSA.

ically this is done by adding a stack alphabet and labeling each transition with a stack operation (a stack symbol to be pushed onto, popped or read from the stack) in additon to the usual input label (Aho and Ullman, 1972; Berstel, 1979) and weight (Kuich and Salomaa, 1986; Petre and Salomaa, 2009). Our equivalent representation allows a transition to be labeled by a stack operation or a regular input symbol but not both. Stack operations are represented by pairs of open and close parentheses (pushing a symbol on and popping it from the stack). The advantage of this representation is that is identical to the finite automaton representation except that certain symbols (the parentheses) have special semantics. As such, several finite-state algorithms either immediately generalize to this PDA representation or do so with minimal changes. The algorithms described in this section have been implemented in the PDT extension (Allauzen and Riley, 2011) of the OpenFst library (Allauzen et al., 2007).

2.1 Definitions

A (restricted) Dyck language consist of "well-formed" or "balanced" strings over a finite number of pairs of parentheses. Thus the string ([( ) ( )] { } [ ]) ( ) is in the Dyck language over 3 pairs of parentheses.

More formally, let A and $\overline{A}$ be two finite alphabets such that there exists a bijection f from A to $\overline{A}$. Intuitively, f maps an open parenthesis to its corresponding close parenthesis. Let $\bar{a}$ denote $f(a)$ if $a \in A$ and $f^{-1}(a)$ if $a \in \overline{A}$. The *Dyck language* D_A over the alphabet $\widehat{A} = A \cup \overline{A}$ is then the language defined by the following context-free grammar: $S \to \epsilon$, $S \to SS$ and $S \to aS\bar{a}$ for all $a \in A$. We define the mapping $c_A : \widehat{A}^* \to \widehat{A}^*$ as follow. $c_A(x)$ is the string obtained by iteratively deleting from x all factors of the form $a\bar{a}$ with $a \in A$. Observe that $D_A = c_A^{-1}(\epsilon)$.

Let A and B be two finite alphabets such that $B \subseteq A$, we define the mapping $r_B : A^* \to B^*$ by $r_B(x_1 \dots x_n) = y_1 \dots y_n$ with $y_i = x_i$ if $x_i \in B$ and $y_i = \epsilon$ otherwise.

A *weighted pushdown automaton* (PDA) T over the tropical semiring $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ is a 9-tuple $(\Sigma, \Pi, \overline{\Pi}, Q, E, I, F, \rho)$ where Σ is the finite input alphabet, Π and $\overline{\Pi}$ are the finite open and close parenthesis alphabets, Q is a finite set of states, $I \in Q$ the initial state, $F \subseteq Q$ the set of final states, $E \subseteq Q \times (\Sigma \cup \widehat{\Pi} \cup \{\epsilon\}) \times (\mathbb{R} \cup \{\infty\}) \times Q$ a finite set of transitions, and $\rho : F \to \mathbb{R} \cup \{\infty\}$ the final weight function. Let $e = (p[e], i[e], w[e], n[e])$ denote a transition in E.

A path π is a sequence of transitions $\pi = e_1 \dots e_n$ such that $n[e_i] = p[e_{i+1}]$ for $1 \leq i < n$. We then define $p[\pi] = p[e_1]$, $n[\pi] = n[e_n]$, $i[\pi] = i[e_1] \cdots i[e_n]$, and $w[\pi] = w[e_1] + \ldots + w[e_n]$.

A path π is accepting if $p[\pi] = I$ and $n[\pi] \in F$. A path π is balanced if $r_{\widehat{\Pi}}(i[\pi]) \in D_\Pi$. A balanced path π accepts the string $x \in \Sigma^*$ if it is a balanced accepting path such that $r_\Sigma(i[\pi]) = x$.

The *weight associated by* T to a string $x \in \Sigma^*$ is $T(x) = \min_{\pi \in P(x)} w[\pi] + \rho(n[\pi])$ where $P(x)$ denotes the set of balanced paths accepting x. A weighted language is recognizable by a weighted pushdown automaton iff it is context-free. We define the *size* of T as $|T| = |Q| + |E|$.

A PDA T has *a bounded stack* if there exists $K \in \mathbb{N}$ such that for any sub-path π of any balanced path in T: $|c_\Pi(r_{\widehat{\Pi}}(i[\pi]))| \leq K$. If T has a bounded stack, then it represents a regular language. Figure 1 shows non-regular, regular and bounded-stack PDAs.

A *weighted finite automaton* (FSA) can be viewed as a PDA where the open and close parentheses alphabets are empty, see (Mohri, 2009) for a standalone definition.

2.2 Expansion Algorithm

Given a bounded-stack PDA T, the *expansion* of T is the FSA T' equivalent to T defined as follows.

A state in T' is a pair (q, z) where q is a state in T and $z \in \Pi^*$. A transition (q, a, w, q') in T results in a transition $((q, z), a', w, (q', z'))$ in T' only when: (a) $a \in \Sigma \cup \{\epsilon\}$, $z' = z$ and $a' = a$, (b) $a \in \Pi$, $z' = za$ and $a' = \epsilon$, or (c) $a \in \overline{\Pi}$, z' is such that $z = z'\overline{a}$ and $a' = \epsilon$. The initial state of T' is $I' = (I, \epsilon)$. A state (q, z) in T' is final if q is final in T and $z = \epsilon$ ($\rho'((q, \epsilon)) = \rho(q)$). The set of states of T' is the set of pairs (q, z) that can be reached from an initial state by transitions defined as above. The condition that T has a bounded stack ensures that this set is finite (since it implies that for any (q, z), $|z| \leq K$).

The complexity of the algorithm is linear in $O(|T'|) = O(e^{|T|})$. Figure 1d show the result of the algorithm when applied to the PDA of Figure 1c.

2.3 Intersection Algorithm

The class of weighted pushdown automata is closed under intersection with weighted finite automata (Bar-Hillel et al., 1964; Nederhof and Satta, 2003). Considering a pair (T_1, T_2) where one element is an FSA and the other element a PDA, then there exists a PDA $T_1 \cap T_2$, the *intersection* of T_1 and T_2, such that for all $x \in \Sigma^*$: $(T_1 \cap T_2)(x) = T_1(x) + T_2(x)$. We assume in the following that T_2 is an FSA. We also assume that T_2 has no input-ϵ transitions. When T_2 has input-ϵ transitions, an epsilon filter (Mohri, 2009; Allauzen et al., 2011) generalized to handle parentheses can be used.

A state in $T = T_1 \cap T_2$ is a pair (q_1, q_2) where q_1 is a state of T_1 and q_2 a state of T_2. The initial state is $I = (I_1, I_2)$. Given a transition $e_1 = (q_1, a, w_1, q_1')$ in T_1, transitions out of (q_1, q_2) in T are obtained using the following rules.

If $a \in \Sigma$, then e_1 can be matched with a transition (q_2, a, w_2, q_2') in T_2 resulting a transition $((q_1, q_2), a, w_1 + w_2, (q_1', q_2'))$ in T.

If $a = \epsilon$, then e_1 is matched with staying in q_2 resulting in a transition $((q_1, q_2), \epsilon, w_1, (q_1', q_2))$.

Finally, if $a \in \widehat{\Pi}$, e_1 is also matched with staying in q_2, resulting in a transition $((q_1, q_2), a, w_1, (q_1', q_2))$ in T.

A state (q_1, q_2) in T is final when both q_1 and q_2 are final, and then $\rho((q_1, q_2)) = \rho_1(q_1) + \rho_2(q_2)$.

```
SHORTESTDISTANCE(T)
 1  for each q ∈ Q and a ∈ Π do
 2     B[q, a] ← ∅
 3  GETDISTANCE(T, I)
 4  return d[f, I]

RELAX(q, s, w, S)
 1  if d[q, s] > w then
 2     d[q, s] ← w
 3     if q ∉ S then
 4        ENQUEUE(S, q)

GETDISTANCE(T, s)
 1  for each q ∈ Q do
 2     d[q, s] ← ∞
 3  d[s, s] ← 0
 4  S_s ← s
 5  while S_s ≠ ∅ do
 6     q ← HEAD(S_s)
 7     DEQUEUE(S_s)
 8     for each e ∈ E[q] do
 9        if i[e] ∈ Σ ∪ {ϵ} then
10           RELAX(n[e], s, d[q, s] + w[e], S_s)
11        elseif i[e] ∈ Π̄ then
12           B[s, ī[e]] ← B[s, ī[e]] ∪ {e}
13        elseif i[e] ∈ Π then
14           if d[n[e], n[e]] is undefined then
15              GETDISTANCE(T, n[e])
16           for each e' ∈ B[n[e], i[e]] do
17              w ← d[q, s] + w[e] + d[p[e'], n[e]] + w[e']
18              RELAX(n[e'], s, w, S_s)
```

Figure 2: PDA shortest distance algorithm. We assume that $F = \{f\}$ and $\rho(f) = 0$ to simplify the presentation.

The complexity of the algorithm is in $O(|T_1||T_2|)$.

2.4 Shortest Distance and Path Algorithms

A *shortest path* in a PDA T is a balanced accepting path with minimal weight and the *shortest distance* in T is the weight of such a path. We show that when T has a bounded stack, shortest distance and shortest path can be computed in $O(|T|^3 \log |T|)$ time (assuming T has no negative weights) and $O(|T|^2)$ space.

Given a state s in T with at least one incoming open parenthesis transition, we denote by C_s the set of states that can be reached from s by a balanced path. If s has several incoming open parenthesis transitions, a naive implementation might lead to the states in C_s to be visited up to exponentially many times. The basic idea of the algorithm is to memoize the shortest distance from s to states in C_s. The

pseudo-code is given in Figure 2.

GETDISTANCE(T, s) starts a new instance of the shortest-distance algorithm from s using the queue $\mathcal{S}_s$, initially containing s. While the queue is not empty, a state is dequeued and its outgoing transitions examined (line 5-9). Transitions labeled by non-parenthesis are treated as in Mohri (2009) (line 9-10). When the considered transition e is labeled by a close parenthesis, it is remembered that it balances all incoming open parentheses in s labeled by $\overline{i[e]}$ by adding e to $B[s, \overline{i[e]}]$ (line 11-12). Finally, when e is labeled with an open parenthesis, if its destination has not already been visited, a new instance is started from $n[e]$ (line 14-15). The destination states of all transitions balancing e are then relaxed (line 16-18).

The space complexity of the algorithm is quadratic for two reasons. First, the number of non-infinity $d[q, s]$ is $|Q|^2$. Second, the space required for storing B is at most in $O(|E|^2)$ since for each open parenthesis transition e, the size of $|B[n[e], i[e]]|$ is $O(|E|)$ in the worst case. This last observation also implies that the cumulated number of transitions examined at line 16 is in $O(N|Q|\ |E|^2)$ in the worst case, where N denotes the maximal number of times a state is inserted in the queue for a given call of GETDISTANCE. Assuming the cost of a queue operation is $\Gamma(n)$ for a queue containing n elements, the worst-case time complexity of the algorithm can then be expressed as $O(N|T|^3\ \Gamma(|T|))$. When T contains no negative weights, using a shortest-first queue discipline leads to a time complexity in $O(|T|^3 \log |T|)$. When all the C_s's are acyclic, using a topological order queue discipline leads to a $O(|T|^3)$ time complexity.

In effect, we are solving a k-sources shortest-path problem with k single-source solutions. A potentially better approach might be to solve the k-sources or k-pairs problem directly (Hershberger et al., 2003).

When T has been obtained by converting an RTN or an hypergraph into a PDA (Section 2.5), the polynomial dependency in $|T|$ becomes a linear dependency both for the time and space complexities. Indeed, for each q in T, there exists a unique s such that $d[q, s]$ is non-infinity. Moreover, for each close parenthesis transistion e, there exists a unique open parenthesis transition e' such that $e \in B[n[e'], i[e']]$. When each component of the RTN is acyclic, the complexity of the algorithm is hence in $O(|T|)$ in time and space.

The algorithm can be modified to compute the shortest path by keeping track of parent pointers.

2.5 Replacement Algorithm

A *recursive transition network* (RTN) can be specified by $(N, \Sigma, (T_\nu)_{\nu \in N}, S)$ where N is an alphabet of nonterminals, Σ is the input alphabet, $(T_\nu)_{\nu \in N}$ is a family of FSAs with input alphabet $\Sigma \cup N$, and $S \in N$ is the root nonterminal.

A string $x \in \Sigma^*$ is accepted by R if there exists an accepting path π in T_S such that recursively replacing any transition with input label $\nu \in N$ by an accepting path in T_ν leads to a path π^* with input x. The weight associated by R is the minimum over all such π^* of $w[\pi^*] + \rho_S(n[\pi^*])$.

Given an RTN R, the *replacement* of R is the PDA T equivalent to R defined by the 9-tuple $(\Sigma, \Pi, \overline{\Pi}, Q, E, I, F, \sigma, \rho)$ with $\Pi = Q = \bigcup_{\nu \in N} Q_\nu$, $I = I_S$, $F = F_S$, $\rho = \rho_S$, and $E = \bigcup_{\nu \in N} \bigcup_{e \in E_\nu} E^e$ where $E^e = \{e\}$ if $i[e] \notin N$ and $E^e = \{(p[e], n[e], w[e], I_\mu), (f, \overline{n[e]}, \rho_\mu(f), n[e]) | f \in F_\mu\}$ with $\mu = i[e] \in N$ otherwise.

The complexity of the construction is in $O(|T|)$. If $|F_\nu| = 1$, then $|T| = O(\sum_{\nu \in N} |T_\nu|) = O(|R|)$. Creating a superfinal state for each T_ν would lead to a T whose size is always linear in the size of R.

3 Hierarchical Phrase-Based Translation Representation

In this section, we compare several different representations for the target translations $\mathcal{T}$ of the source sentence s by synchronous CFG G prior to language model M application. As discussed in the introduction, $\mathcal{T}$ is a context-free language. For example, suppose it corresponds to:

$$S \rightarrow abXdg,\ S \rightarrow acXfg,\ \text{and}\ X \rightarrow bc.$$

Figure 3 shows several alternative representations of $\mathcal{T}$: Figure 3a shows the hypergraph representation of this grammar; there is a 1:1 correspondence between each production in the CFG and each hyperedge in the hypergraph. Figure 3b shows the RTN representation of this grammar with a 1:1 correspondence between each production in the CFG and each path in the RTN; this is the translation representation pro-

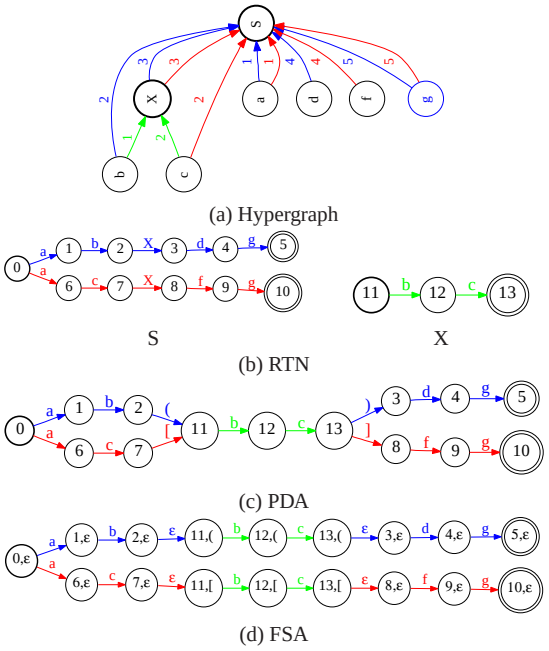


Figure 3: Alternative translation representations

duced by the HiFST decoder (Iglesias et al., 2009a; de Gispert et al., 2010). Figure 3c shows the push-down automaton representation generated from the RTN with the replacement algorithm of Section 2.5. Since s is a finite language and G does not allow unbounded insertion, T_p has a bounded stack and $\mathcal{T}$ is, in fact, a regular language. Figure 3d shows the finite-state automaton representation of $\mathcal{T}$ generated by the PDA using the expansion algorithm of Section 2.2. The HiFST decoder converts its RTN translation representation immediately into the finite-state representation using an algorithm equivalent to converting the RTN into a PDA followed by PDA expansion.

As shown in Figure 4, an advantage of the RTN, PDA, and FSA representations is that they can benefit from FSA epsilon removal, determinization and minimization algorithms applied to their components (for RTNs and PDAs) or their entirety (for FSAs). For the complexity discussion below, however, we disregard these optimizations. Instead we focus on the complexity of each MT step described in the introduction:

1. *SCFG Translation:* Assuming that the parsing of the input is performed by a CYK parse, then the CFG, hypergraph, RTN and PDA representations can be generated in $O(|s|^3|G|)$ time and space (Aho and Ullman, 1972). The FSA representation can require an additional $O(e^{|s|^3|G|})$ time and space since the PDA expansion can be exponential.
2. *Intersection:* The intersection of a CFG T_h with a finite automaton M can be performed by the classical Bar-Hillel algorithm (Bar-Hillel et al., 1964) with time and space complexity $O(|T_h||M|^3)$.[1] The PDA intersection algorithm from Section 2.3 has time and space complexity $O(|T_p||M|)$. Finally, the FSA intersection algorithm has time and space complexity $O(|T_f||M|)$ (Mohri, 2009).
3. *Shortest Path:* The shortest path algorithm on the hypergraph, RTN, and FSA representations requires linear time and space (given the underlying acyclicity) (Huang, 2008; Mohri, 2009). As presented in Section 2.4, the PDA representation can require time cubic and space quadratic in $|M|$.[2]

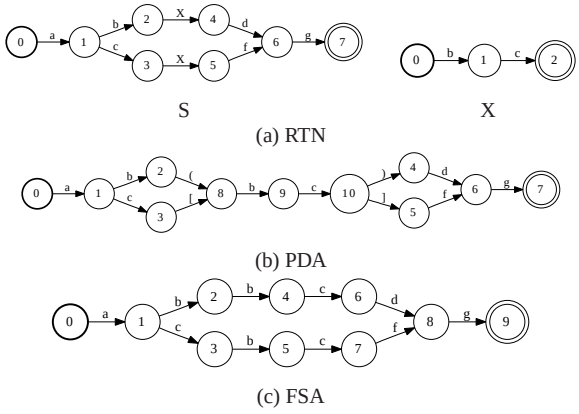


Figure 4: Optimized translation representations

Table 1 summarizes the complexity results. Note the PDA representation is equivalent in time and superior in space to the CFG/hypergraph representation, in general, and it can be superior in both space

[1]The modified Bar-Hillel construction described by Chiang (2007) has time and space complexity $O(|T_h||M|^4)$; the modifications were introduced presumably to benefit the subsequent pruning method employed (but see Huang et al. (2005)).

[2]The time (resp. space) complexity is not cubic (resp. quadratic) in $|T_p||M|$. Given a state q in T_p, there exists a unique s_q such that q belongs to C_{s_q}. Given a state (q_1, q_2) in $T_p \cap M$, $(q_1, q_2) \in C_{(s_1,s_2)}$ only if $s_1 = s_{q_1}$, and hence (q_1, q_2) belongs to at most $|M|$ components.

Representation	Time Complexity	Space Complexity
CFG/hypergraph	$O(\|s\|^3 \, \|G\| \, \|M\|^3)$	$O(\|s\|^3 \, \|G\| \, \|M\|^3)$
PDA	$O(\|s\|^3 \, \|G\| \, \|M\|^3)$	$O(\|s\|^3 \, \|G\| \, \|M\|^2)$
FSA	$O(e^{\|s\|^3 \|G\|} \, \|M\|)$	$O(e^{\|s\|^3 \|G\|} \, \|M\|)$

Table 1: Complexity using various target translation representations.

and time to the FSA representation depending on the relative SCFG and LM sizes. The FSA representation favors smaller target translation sets and larger language models. Should a better complexity PDA shortest path algorithm be found, this conclusion could change. In practice, the PDA and FSA representations benefit hugely from the optimizations mentioned above, these optimizations improve the time and space usage by one order of magnitude.

4 Experimental Framework

We use two hierarchical phrase-based SMT decoders. The first one is a lattice-based decoder implemented with weighted finite-state transducers (de Gispert et al., 2010) and described in Section 3. The second decoder is a modified version using PDAs as described in Section 2. In order to distinguish both decoders we call them HiFST and HiPDT, respectively. The principal difference between the two decoders is where the finite-state *expansion* step is done. In HiFST, the RTN representation is immediately expanded to an FSA. In HiPDT, this expansion is delayed as late as possible - in the output of the shortest path algorithm. Another possible configuration is to expand after the LM intersection step but before the shortest path algorithm; in practice this is quite similar to HiFST.

In the following sections we report experiments in Chinese-to-English translation. For translation model training, we use a subset of the GALE 2008 evaluation parallel text;[3] this is 2.1M sentences and approximately 45M words per language. We report translation results on a development set *tune-nw* (1,755 sentences) and a test set *test-nw* (1,671 sentences). These contain translations produced by the GALE program and portions of the newswire sections of MT02 through MT06. In tuning the systems, standard MERT (Och, 2003) iterative parameter estimation under IBM BLEU[4] is performed on the development set.

[3]See http://projects.ldc.upenn.edu/gale/data/catalog.html. We excluded the UN material and the LDC2002E18, LDC2004T08, LDC2007E08 and CUDonga collections.

$\mathbf{0}$	$\mathbf{7.5 \times 10^{-9}}$	$\mathbf{7.5 \times 10^{-8}}$	$\mathbf{7.5 \times 10^{-7}}$
207.5	20.2	4.1	0.9

Table 2: Number of ngrams (in millions) in the 1st pass 4-gram language models obtained with different θ values (top row).

The parallel corpus is aligned using MTTK (Deng and Byrne, 2008) in both source-to-target and target-to-source directions. We then follow standard heuristics (Chiang, 2007) and filtering strategies (Iglesias et al., 2009b) to extract hierarchical phrases from the union of the directional word alignments. We call a translation grammar the set of rules extracted from this process. We extract two translation grammars:

- A restricted grammar where we apply the following additional constraint: rules are only considered if they have a forward translation probability $p > 0.01$. We call this G_1. As will be discussed later, the interest of this grammar is that decoding under it can be exact, that is, without any pruning in search.
- An unrestricted one without the previous constraint. We call this G_2. This is a superset of the previous grammar, and exact search under it is not feasible for HiFST: pruning is required in search.

The initial English language model is a Kneser-Ney 4-gram estimated over the target side of the parallel text and the AFP and Xinhua portions of monolingual data from the English Gigaword Fourth Edition (LDC2009T13). This is a total of 1.3B words. We will call this language model M_1. For large language model rescoring we also use the LM M_2 obtained by interpolating M_1 with a zero-cutoff stupid-backoff (Brants et al., 2007) 5-gram estimated using 6.6B words of English newswire text.

We next describe how we build translation systems using entropy-pruned language models.

1. We build a baseline HiFST system that uses M_1 and a hierarchical grammar G, parameters being optimized with MERT under BLEU.

[4]See ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13.pl

2. We then use entropy-based pruning of the language model (Stolcke, 1998) under a relative perplexity threshold of θ to reduce the size of M_1. We will call the resulting language model as M_1^θ. Table 2 shows the number of n-grams (in millions) obtained for different θ values.
3. We translate with M_1^θ using the same parameters obtained in MERT in step 1, except for the word penalty, tuned over the lattices under BLEU performance. This produces a translation lattice in the topmost cell that contains hypotheses with exact scores under the translation grammar and M_1^θ.
4. Translation lattices in the topmost cell are pruned with a likelihood-based beam width β.
5. We remove the M_1^θ scores from the pruned translation lattices and reapply M_1, moving the word penalty back to the original value obtained in MERT. These operations can be carried out efficiently via standard FSA operations.
6. Additionally, we can rescore the translation lattices obtained in steps 1 or 5 with the larger language model M_2. Again, this can be done via standard FSA operations.

Note that if $\beta = \infty$ or if $\theta = 0$, the translation lattices obtained in step 1 should be identical to the ones of step 5. While the goal is to increase θ to reduce the size of the language model used at Step 3, β will have to increase accordingly so as to avoid pruning away desirable hypotheses in Step 4. If β defines a sufficiently wide beam to contain the hypotheses which would be favoured by M_1, faster decoding with M_1^θ would be possible without incurring search errors M_1. This is investigated next.

5 Entropy-Pruned LM in Rescoring

In Table 3 we show translation performance under grammar G_1 for different values of θ. Performance is reported after first-pass decoding with M_1^θ (see step 3 in Section 4), after rescoring with M_1 (see step 5) and after rescoring with M_2 (see step 6). The baseline (experiment number 1) uses $\theta = 0$ (that is, M_1) for decoding.

Under translation grammar G_1, HiFST is able to generate an FSA with the entire space of possible candidate hypotheses. Therefore, any degradation in performance is only due to the M_1^θ involved in decoding and the β applied prior to rescoring.

As shown in row number 2, for $\theta \leq 10^{-9}$ the system provides the same performance to the baseline when $\beta > 8$, while decoding time is reduced by roughly 40%. This is because M_1^θ is 10% of the size of the original language model M_1, as shown in Table 2. As M_1^θ is further reduced by increasing θ (see rows number 3 and 4), decoding time is also reduced. However, the beam width β required in order to recover the good hypotheses in rescoring increases, reaching 12 for experiment 3 and 15 for experiment 4.

Regarding rescoring with the larger M_2 (step 6 in Section 4), the system is also able to match the baseline performance as long as β is wide enough, given the particular M_1^θ used in first-pass decoding. Interestingly, results show that a similar β value is needed when rescoring either with M_1 or M_2.

The usage of entropy-pruned language models increments speed at the risk of search errors. For instance, comparing the outputs of systems 1 and 2 with $\beta = 10$ in Table 3 we find 45 different 1-best hypotheses, even though the BLEU score is identical. In other words, we have 45 cases in which system 2 is not able to recover the baseline output because the 1st-pass likelihood beam β is not wide enough. Similarly, system 3 fails in 101 cases ($\beta = 12$) and system 4 fails in 95 cases. Interestingly, some of these sentences would require impractically huge beams. This might be due to the Kneser-Ney smoothing, which interacts badly with entropy pruning (Chelba et al., 2010).

6 Hiero with PDAs and FSAs

In this section we contrast HiFST with HiPDT under the same translation grammar and entropy-pruned language models. Under the constrained grammar G_1 their performance is identical as both decoders can generate the entire search space which can then be rescored with M_1 or M_2 as shown in the previous section.

Therefore, we now focus on the unconstrained grammar G_2, where exact search is not feasible for HiFST. In order to evaluate this problem, we run both decoders over *tune-nw*, restricting memory usage to 10 gigabytes. If this limit is reached in decod-

HiFST (G_1 + M_1^θ)						+M_1		+M_2	
#	θ	*tune-nw*	*test-nw*	*time*	β	*tune-nw*	*test-nw*	*tune-nw*	*test-nw*
1	**0 (M_1)**	**34.3**	**34.5**	**0.68**	-	-	-	**34.8**	**35.6**
2	7.5×10^{-9}	32.0	32.8	0.38	*10*	34.3	34.5	34.8	35.6
					9			34.9	35.5
					8				
3	7.5×10^{-8}	29.5	30.0	0.28	*12*	34.2	34.5	34.7	35.6
					9	34.3	34.4	34.8	35.2
					8	34.2			35.1
4	7.5×10^{-7}	26.0	26.4	0.20	*15*	34.2	34.5	34.7	35.6
					12		34.4		35.5

Table 3: Results (lowercase IBM BLEU scores) under G_1 with various M_1^θ as obtained with several values of θ. Performance in subsequent rescoring with M_1 and M_2 after likelihood-based pruning of the translation lattices for various β is also reported. Decoding time, in seconds/word over *test-nw*, refers strictly to first-pass decoding.

Exact search for G_2 + M_1^θ with memory usage under 10 GB							
#	θ	HiFST			HiPDT		
		Success	**Failure**		**Success**	**Failure**	
			Expand	*Compose*		*Compose*	*Expand*
2	7.5×10^{-9}	12	51	37	40	8	52
3	7.5×10^{-8}	16	53	31	76	1	23
4	7.5×10^{-7}	18	53	29	99.8	0	0.2

Table 4: Percentage of success in producing the 1-best translation under G_2 with various M_1^θ when applying a hard memory limitation of 10 GB, as measured over *tune-nw* (1755 sentences). If decoder fails, we report what step was being done when the limit was reached. HiFST could be expanding into an FSA or composing the FSA with M_1^θ; HiPDT could be PDA composing with M_1^θ or PDA expanding into an FSA.

HiPDT (G_2 + M_1^θ)				+M_1		+M_2	
θ	*tune-nw*	*test-nw*	β	*tune-nw*	*test-nw*	*tune-nw*	*test-nw*
7.5×10^{-7}	25.7	26.3	15	34.6	34.8	35.2	36.1

Table 5: HiPDT performance on grammar G_2 with $\theta = 7.5 \times 10^{-7}$. Exact search with HiFST is not possible under these conditions: pruning during search would be required.

ing, the process is killed[5]. We report what internal decoding operation caused the system to crash. For HiFST, these include expansion into an FSA (*Expand*) and subsequent intersection with the language model (*Compose*). For HiPDT, these include PDA intersection with the language model (*Compose*) and subsequent expansion into an FSA (*Expand*), using algorithms described in Section 2.

Table 4 shows the number of times each decoder succeeds in finding a hypothesis given the memory limit, and the operations being carried out when they fail to do so, when decoding with various M_1^θ. With $\theta = 7.5 \times 10^{-9}$ (row 2), HiFST can only decode 218 sentences, while HiPDT succeeds in 703 cases. The differences between both decoders increase as the M_1^θ is more reduced, and for $\theta = 7.5 \times 10^{-7}$ (row 4), HiPDT is able to perform exact search over all but three sentences.

Table 5 shows performance using the latter configuration (Table 4, row 4). After large language model rescoring, HiPDT improves 0.5 BLEU over baseline with G_1 (Table 3, row 1).

7 Discussion and Conclusion

HiFST fails to decode mainly because the expansion into an FST leads to far too big search spaces (e.g. fails 938 times under $\theta = 7.5 \times 10^{-8}$). If it succeeds in expanding the search space into an FST, the decoder still has to compose with the language model, which is also critical in terms of memory us-

[5]We used *ulimit* command. The experiment was carried out over machines with different configurations and load. Therefore, these numbers must be considered as approximate values.

age (fails 536 times). In contrast, HiPDT creates a PDA, which is a more compact representation of the search space and allows efficient intersection with the language model before expansion into an FST. Therefore, the memory usage is considerably lower. Nevertheless, the complexity of the language model is critical for the PDA intersection and very specially the PDA expansion into an FST (fails 403 times for $\theta = 7.5 \times 10^{-8}$).

With the algorithms presented in this paper, decoding with PDAs is possible for any translation grammar as long as an entropy pruned LM is used. While this allows exact decoding, it comes at the cost of making decisions based on less complex LMs, although this has been shown to be an adequate strategy when applying compact CFG rulesets. On the other hand, HiFST cannot decode under large translation grammars, thus requiring pruning during lattice construction, but it can apply an unpruned LM in this process. We find that with carefully designed pruning strategies, HiFST can match the performance of HiPDT reported in Table 5. But without pruning in search, expansion directly into an FST would lead to an explosion in terms of memory usage. Of course, without memory constraints both strategies would reach the same performance.

Overall, these results suggest that HiPDT is more robust than HiFST when using complex hierarchical grammars. Conversely, FSTs might be more efficient for search spaces described by more constrained hierarchical grammars. This suggests that a hybrid solution could be effective: we could use PDAs or FSTs e.g. depending on the number of states of the FST representing the expanded search space, or other conditions.

8 Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7-ICT-2009-4) under grant agreement number 247762, and was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No.HR0011-06-C-0022, and a Google Faculty Research Award, May 2010.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1-2. Prentice-Hall.

Cyril Allauzen and Michael Riley, 2011. *Pushdown Transducers*. http://pdt.openfst.org.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of CIAA*, pages 11–23. http://www.openfst.org.

Cyril Allauzen, Michael Riley, and Johan Schalkwyk. 2011. Filters for efficient composition of weighted finite-state transducers. In *Proceedings of CIAA*, volume 6482 of *LNCS*, pages 28–38. Springer.

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

Jean Berstel. 1979. *Transductions and Context-Free Languages*. Teubner.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-ACL*, pages 858–867.

Ciprian Chelba, Thorsten Brants, Will Neveitt, and Peng Xu. 2010. Study on interaction between entropy pruning and kneser-ney smoothing. In *Proceedings of Interspeech*, pages 2242–2245.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. 2010. Hierarchical phrase-based translation with weighted finite state transducers and shallow-n grammars. *Computational Linguistics*, 36(3).

Yonggang Deng and William Byrne. 2008. HMM word and phrase alignment for statistical machine translation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):494–507.

Manfred Drosde, Werner Kuick, and Heiko Vogler, editors. 2009. *Handbook of Weighted Automata*. Springer.

John Hershberger, Subhash Suri, and Amit Bhosle. 2003. On the difficulty of some shortest path problems. In *Proceedings of STACS*, volume 2607 of *LNCS*, pages 343–354. Springer.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of EMNLP*, pages 273–283.

Liang Huang, Hao Zhang, and Daniel Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 65–73, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liang Huang. 2008. Advanced dynamic programming in semiring and hypergraph frameworks. In *Proceedings of COLING*, pages 1–18.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009a. Hierarchical phrase-based translation with weighted finite state transducers. In *Proceedings of NAACL-HLT*, pages 433–441.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009b. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of EACL*, pages 380–388.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298.

Werner Kuich and Arto Salomaa. 1986. *Semirings, automata, languages*. Springer.

Shankar Kumar, Yonggang Deng, and William Byrne. 2006. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1):35–75.

Andrej Ljolje, Fernando Pereira, and Michael Riley. 1999. Efficient general lattice generation and rescoring. In *Proceedings of Eurospeech*, pages 1251–1254.

Mehryar Mohri. 2009. Weighted automata algorithms. In Drosde et al. (Drosde et al., 2009), chapter 6, pages 213–254.

Mark-Jan Nederhof and Giorgio Satta. 2003. Probabilistic parsing as intersection. In *Proceedings of 8th International Workshop on Parsing Technologies*, pages 137–148.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Ion Petre and Arto Salomaa. 2009. Algebraic systems and pushdown automata. In Drosde et al. (Drosde et al., 2009), chapter 7, pages 257–289.

R. Prasad, K. Krstovski, F. Choi, S. Saleem, P. Natarajan, M. Decerbo, and D. Stallard. 2007. Real-time speech-to-speech translation for pdas. In *Proceedings of IEEE International Conference on Portable Information Devices*, pages 1 –5.

Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through lagrangian relaxation. In *Proceedings of ACL-HLT*, pages 72–82.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Improved Transliteration Mining Using Graph Reinforcement

Ali El-Kahky[1], Kareem Darwish[1], Ahmed Saad Aldein[2], Mohamed Abd El-Wahab[3], Ahmed Hefny[2], Waleed Ammar[4]

[1] Qatar Computing Research Institute, Qatar Foundation, Doha, Qatar
[2] Computer Engineering Department, Cairo University, Cairo, Egypt
[3] Microsoft Research, Microsoft, Cairo, Egypt
[4] Microsoft Research, Microsoft, Redmond, WA, US

{aelkahky,kdarwish}@qf.org.qa[1], asaadaldien@hotmail.com[2], ahmed.s.hefny@gmail.com[2], t-momah@microsoft.com[3], i-waamma@microsoft.com[4]

Abstract

Mining of transliterations from comparable or parallel text can enhance natural language processing applications such as machine translation and cross language information retrieval. This paper presents an enhanced transliteration mining technique that uses a generative graph reinforcement model to infer mappings between source and target character sequences. An initial set of mappings are learned through automatic alignment of transliteration pairs at character sequence level. Then, these mappings are modeled using a bipartite graph. A graph reinforcement algorithm is then used to enrich the graph by inferring additional mappings. During graph reinforcement, appropriate link reweighting is used to promote good mappings and to demote bad ones. The enhanced transliteration mining technique is tested in the context of mining transliterations from parallel Wikipedia titles in 4 alphabet-based languages pairs, namely English-Arabic, English-Russian, English-Hindi, and English-Tamil. The improvements in F1-measure over the baseline system were 18.7, 1.0, 4.5, and 32.5 basis points for the four language pairs respectively. The results herein outperform the best reported results in the literature by 2.6, 4.8, 0.8, and 4.1 basis points for the four language pairs respectively.

Introduction

Transliteration Mining (TM) is the process of finding transliterated word pairs in parallel or comparable corpora. TM has many potential applications such as mining training data for transliteration, improving lexical coverage for machine translation, and cross language retrieval via translation resource expansion. TM has been gaining some attention lately with a shared task in the ACL 2010 NEWS workshop (Kumaran, et al. 2010).

One popular statistical TM approach is performed in two stages. First, a generative model is trained by performing automatic character level alignment of parallel transliterated word pairs to find character segment mappings between source and target languages. Second, given comparable or parallel text, the trained generative model is used to generate possible transliterations of a word in the source language while constraining the transliterations to words that exist in the target language.

However, two problems arise in this approach:

1. Many possible character sequence mappings between source and target languages may not be observed in training data, particularly when limited training data is available – hurting recall.
2. Conditional probability estimates of obtained mappings may be inaccurate, because some mappings and some character sequences may not

appear a sufficient number of times in training to properly estimate their probabilities – hurting precision.

In this paper we focus on overcoming these two problems to improve overall TM. To address the first problem, we modeled the automatically obtained character sequence mappings (from alignment) as a bipartite graph and then we performed graph reinforcement to enrich the graph and predict possible mappings that were not directly obtained from training data. The example in Figure 1 motivates graph reinforcement. In the example, the Arabic letter "ق" (pronounced as "qa") was not aligned to the English letter "c" in training data. Such a mapping seems probable given that another Arabic letter, "ك" (pronounced as "ka"), maps to two English letters, "q" and "k", to which "ق" also maps. In this case, there are multiple paths that would lead to a mapping between the Arabic letter "ق" and the English letter "c", namely ق ➔ q ➔ ك ➔ c and ق ➔ k ➔ ك ➔ c. By using multiple paths as sources of evidence, we can infer the new mapping and estimate its probability.

Another method for overcoming the missing mappings problem entails assigning small smoothing probabilities to unseen mappings. However, from looking at the graph, it is evident that some mappings could be inferred and should be assigned probabilities that are higher than a small smoothing probability.

The second problem has to do primarily with some characters in one language, typically vowels, mapping to many character sequences in the other language, with some of these mappings assuming very high probabilities (due to limited training data). To overcome this problem, we used link reweighting in graph reinforcement to scale down the likelihood of mappings to target character sequences in proportion to how many source sequences map to them.

We tested the proposed method using the ACL 2010 NEWS workshop data for English-Arabic, English-Russian, English-Hindi, and English-Tamil (Kumaran et al., 2010). For each language pair, the standard ACL 2010 NEWS workshop data contained a base set of 1,000 transliteration pairs for training, and set of 1,000 parallel Wikipedia titles for testing.

The contributions of the paper are:

1. Employing graph reinforcement to improve the coverage of automatically aligned data – as they apply to transliteration mining. This positively affects recall.
2. Applying link reweighting to overcome situations where certain tokens – character sequences in the case of transliteration – tend to have many mappings, which are often erroneous. This positively affects precision.

The rest of the paper is organized as follows: Section 2 surveys prior work on transliteration mining; Section 3 describes the baseline TM approach and reports on its effectiveness; Section 4 describes the proposed graph reinforcement along with link reweighting and reports on the observed improvements; and Section 5 concludes the paper.

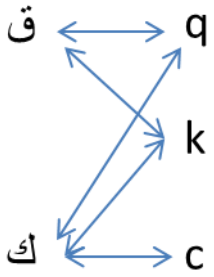


Figure 1: Example mappings seen in training

Background

Much work has been done on TM for different language pairs such as English-Chinese (Kuo et al., 2006; Kuo et al., 2007; Kuo et al., 2008; Jin et al. 2008;), English-Tamil (Saravanan and Kumaran, 2008; Udupa and Khapra, 2010), English-Korean (Oh and Isahara, 2006; Oh and Choi, 2006), English-Japanese (Qu et al., 2000; Brill et al., 2001; Oh and Isahara, 2006), English-Hindi (Fei et al., 2003; Mahesh and Sinha, 2009), and English-Russian (Klementiev and Roth, 2006).

TM typically involves two main tasks, namely: finding character mappings between two languages, and given the mappings ascertaining whether two words are transliterations or not. When training with a limited number of transliteration pairs, two additional problems appear: many possible character sequence mappings between source and target languages may not be observed in training data, and conditional probability estimates of obtained

mappings may be inaccurate. These two problems affect recall and precision respectively.

1.1 Finding Character Mappings

To find character sequence mappings between two languages, the most common approach entails using automatic letter alignment of transliteration pairs. Akin to phrasal alignment in machine translation, character sequence alignment is treated as a word alignment problem between parallel sentences, where transliteration pairs are treated as if they are parallel sentences and the characters from which they are composed are treated as if they are words. Automatic alignment can be performed using different algorithms such as the EM algorithm (Kuo et al., 2008; Lee and Chang, 2003) or HMM based alignment (Udupa et al., 2009a; Udupa et al., 2009b). In this paper, we use automatic character alignment between transliteration pairs using an HMM aligner. Another method is to use automatic speech recognition confusion tables to extract phonetically equivalent character sequences to discover monolingual and cross lingual pronunciation variations (Kuo and Yang, 2005). Alternatively, letters can be mapped into a common character set using a predefined transliteration scheme (Oh and Choi, 2006).

1.2 Transliteration Mining

For the problem of ascertaining if two words can be transliterations of each other, a common approach involves using a generative model that attempts to generate all possible transliterations of a source word, given the character mappings between two languages, and restricting the output to words in the target language (Fei et al., 2003; Lee and Chang, 2003, Udupa et al., 2009a). This is similar to the baseline approach that we used in this paper. Noeman and Madkour (2010) implemented this technique using a finite state automaton by generating all possible transliterations along with weighted edit distance and then filtered them using appropriate thresholds and target language words. They reported the best TM results between English and Arabic with F1-measure of 0.915 on the ACL-2010 NEWS workshop standard TM dataset. A related alternative is to use back-transliteration to determine if one sequence could have been generated by successively mapping character sequences from one language into another (Brill et al., 2001; Bilac and Tanaka, 2005; Oh and Isahara, 2006).

Udupa and Khapra (2010) proposed a method in which transliteration candidates are mapped into a "low-dimensional common representation space". Then, similarity between the resultant feature vectors for both candidates can be computed. Udupa and Kumar (2010) suggested that mapping to a common space can be performed using context sensitive hashing. They applied their technique to find variant spellings of names.

Jiampojamarn et al. (2010) used classification to determine if a source language word and target language word are valid transliterations. They used a variety of features including edit distance between an English token and the Romanized versions of the foreign token, forward and backward transliteration probabilities, and character n-gram similarity. They reported the best results for Russian, Tamil, and Hindi with F1-measure of 0.875, 0.924, and 0.914 respectively on the ACL-2010 NEWS workshop standard TM datasets.

1.3 Training with Limited Training Data

When only limited training data is available to train a character mapping model, the resultant mappings are typically incomplete (due to sparseness in the training data). Further, resultant mappings may not be observed a sufficient of times and hence their mapping probabilities may be inaccurate.

Different methods were proposed to solve these two problems. These methods focused on making training data less sparse by performing some kind of letter conflation. Oh and Choi (2006) used a SOUNDEX like scheme. SOUNDEX is used to convert English words into a simplified phonetic representation, in which vowels are removed and phonetically similar characters are conflated. A variant of SOUNDEX along with iterative training was proposed by Darwish (2010). Darwish (2010) reported significant improvements in TM recall at the cost of limited drop in precision. Another method involved expanding character sequence maps by automatically mining transliteration pairs and then aligning these pairs to generate an expanded set of character sequence maps (Fei et al., 2003). In this work we proposed graph

reinforcement with link reweighting to address this problem. Graph reinforcement was used in the context of different problems such as mining paraphrases (Zhao et al., 2008; Kok and Brockett, 2010; Bannard and Callison-Burch 2005) and named entity translation extraction (You et al., 2010).

Baseline Transliteration Mining

1.4 Description of Baseline System

The basic TM setup that we employed in this work used a generative transliteration model, which was trained on a set of transliteration pairs. The training involved automatically aligning character sequences. The alignment was performed using a Bayesian learner that was trained on word dependent transition models for HMM based word alignment (He, 2007). Alignment produced mappings of source character sequences to target character sequences along with the probability of source given target and vice versa. Source character sequences were restricted to be 1 to 3 characters long.

For all the work reported herein, given an English-foreign language transliteration candidate pair, English was treated as the target language and the foreign language as the source. Given a foreign source language word sequence F_1^n and an English target word sequence E_1^m, $F_i \in F_1^n$ could be a potential transliteration of $E_j \in E_1^m$. An example of word sequences pair is the Tamil-English pair: (முதலாம் ஹைலி செலாசி, Haile Selassie I of Ethiopia), where "முதலாம்" could be transliteration for any or none of the English words {"Haile", "Selassie", "I", "of", "Ethiopia"}. The pseudo code below describes how transliteration mining generates candidates.

Basically, given a source language word, all possible segmentations, where each segment has a maximum length of 3 characters, are produced along with their associated mappings into the target language. Given all mapping combinations, combinations producing valid target words are retained and sorted according to the product of their mapping probabilities. If the product of the mapping probabilities for the top combination is above a certain threshold, then it is chosen as the transliteration candidate. Otherwise, no candidate is chosen. To illustrate how TM works, consider the following example: Given the Arabic word "من", all possible segmentations are (م ، ن) and (من). Given the target words {the, best, man} and the possible mappings for the segments and their probabilities:

مـ = {(m, 0.7), (me, 0.25), (ma, 0.05)}

ن = {n, 0.7), (nu, 0.2), (an, 0.1)}

من = {(men, 0.4), (man, 0.3), (mn, 0.3)}

The only combinations leading valid target words would be:

(من) ➔ {(man: 0.3)}

(مـ ، ن) ➔ {(m,an: 0.07), (ma, n: 0.035)}

Consequently, the algorithm would produce the tuple with the highest probability: (من , man, 0.3).

As the pseudo code suggests, the actual implementation is optimized via: incremental left to right processing of source words; the use of a Patricia trie to prune mapping combinations that don't lead to valid words; and the use of a priority queue to insure that the best candidate is always at the top of the queue.

1.5 Smoothing and Thresholding

We implemented the baseline system with and without assigning small smoothing probabilities for unseen source character to target character mappings. Subsequent to training, the smoothing probability was selected as the smallest observed mapping probability in training.

We used a threshold on the minimum acceptable transliteration score to filter out unreliable transliterations. We couldn't fix a minimum score for reliable transliterations to a uniform value for all words, because this would have caused the model to filter out long transliterations. Thus, we tied the threshold to the length of transliterated words. We assumed a threshold d for single character mappings and the transliteration threshold for a target word of length l was computed as d^l. We selected d by sorting the mapping probabilities, removing the lowest 10% of mapping probabilities (which we assumed to be outliers), and then selecting the smallest observed probability to be the character threshold d. The choice of removing the lowest ranking 10% of mapping probabilities was based on intuition, because we did not have a validation set. The threshold was then applied to filter out transliteration with $TransliterationScore < d^l$.

1: **Input:** *Mappings*, set of source given target mappings with associated *Prob*.
2: **Input:** *SourceWord* ($F_i \in F_1^n$), Source language word
3: **Input:** *TargetWords*, Patricia trie containing all target language words (E_1^m)
4: **Data Structures:** *DFS*, Priority queue to store candidate transliterations pair ordered by their transliteration score – Each candidate transliteration tuple = (*SourceFragment*, *TargetTransliteration*, *TransliterationScore*).
5: *StartSymbol* = ("", "", 1.0)
6: *DFS*={*StartSymbol*}
7: **While**(*DFS is not empty*)
8: *SourceFragment= DFS*.Top().SourceFragment
9: *TargetFragment= DFS*.Top().TargetTransliteration
10: *FragmentProb=DFS*.Top().TransliterationScore
11: **If** (*SourceWord == SourceFragment*)
12: **If**(*FragmentScore > Threshold*)
13: **Return** (*SourceWord, TargetTransliteration, TransliterationScore*)
14: **Else**
15: **Return** *Null*
16: *DFS*.RemoveTop()
17: **For** *SubFragmentLength*=1 to 3
18: *SourceSubString*= SubString(*SourceWord, SourceFragment*.Length , *SubFragmentLength*)
19: **Foreach** *mapping* in *Mappings[SourceSubString]*
20: **If**((*TargetFragment + mapping*) is a sub-string in *TargetWords*)
21: *DFS*.Add(*SourceFragment + SourceSubString, Mapping.Score * FragmentScore*)
22: *DFS*.Remove(SourceFragment)
23: **End While**
24: **Return** *Null*

Figure 2: Pseudo code for transliteration mining

1.6 Effectiveness of Baseline System

To test the effectiveness of the baseline system, we used the standard TM training and test datasets from the ACL-2010 NEWS workshop shared task. The datasets are henceforth collectively referred to as the NEWS dataset. The dataset included 4 alphabet-based language pairs, namely English-Arabic, English-Russian, English-Hindi, and English-Tamil. For each pair, a dataset included a list of 1,000 parallel transliteration word pairs to train a transliteration model, and a list of 1,000 parallel word sequences to test TM. The parallel sequences in the test sets were extracted titles from Wikipedia article for which cross language links exist between both languages.

We preprocessed the different languages as follows:

- Russian: characters were case-folded
- Arabic: the different forms of alef (alef, alef maad, alef with hamza on top, and alef with hamza below it) were normalized to alef, ya and alef maqsoura were normalized to ya, and ta marbouta was mapped to ha.
- English: letters were case-folded and the following letter conflations were performed:
  ž, ż → z á, â, ä, à, ã, ā, ą, æ → a
  é, ę, è → e ć, č, ç →c
  ł →l ï, í, ì, î → i
  ó, ō, ö, õ → o ń, ñ, ṅ → n
  ș, ś, ß, š → s ř → r
  ý → y ū, ü, ú, û → u
- Tamil and Hindi: no preprocessing was performed.

English/	P		R		F	
Arabic	0.988	0.983	0.583	0.603	0.733	0.748
Russian	0.975	0.967	0.831	0.862	0.897	0.912
Hindi	0.986	0.981	0.693	0.796	0.814	0.879
Tamil	0.984	0.981	0.274	0.460	0.429	0.626

Table 1: Baseline results for all language pairs. Results with smoothing are shaded.

Table 1 reports the precision, recall, and F1-measure results for using the baseline system in TM between English and each of the 4 other languages in the NEWS dataset with and without smoothing. As is apparent in the results, without smoothing, precision is consistently high for all languages, but recall is generally poor, particularly for Tamil. When smoothing is applied, we observed a slight drop in precision for Arabic, Hindi, and Tamil and a significant drop of 5.6

basis points for Russian. However, the application of smoothing increased recall dramatically for all languages, particularly Tamil. For the remainder of the paper, the results with smoothing are used as the baseline results.

Background

1.7 Description of Graph Reinforcement

In graph reinforcement, the mappings deduced from the alignment process were represented using a bipartite graph G = (S, T, M), where S was the set of source language character sequences, T was the set of target language character sequences, and M was the set of mappings (links or edges) between S and T. The score of each mapping m(v1|v2), where m(v1|v2) ∈ M, was initially set to the conditional probability of target given source p(v1|v2). Graph reinforcement was performed by traversing the graph from S ➔ T ➔ S ➔ T in order to deduce new mappings. Given a source sequence s' ∈ S and a target sequence t' ∈ T, the deduced mapping probabilities were computed as follows (Eq.1):

$$m(t'|s') = 1 - \prod_{\forall s \in S, t \in T} \left(1 - m(t'|s)m(s|t)m(t|s')\right)$$

where the term $\left(1 - m(t'|s)m(s|t)m(t|s')\right)$ computed the probability that a mapping is not correct. Hence, the probability of an inferred mapping would be boosted if it was obtained from multiple paths. Given the example in Figure 1, m(c|ق) would be computed as follows:

$$1 - \left(1 - m(c|ك)m(ك|q)m(q|ق)\right)\left(1 - m(c|ك)m(ك|k)m(k|ق)\right)$$

We were able to apply reinforcement iteratively on all mappings from S to T to deduce previously unseen mappings (graph edges) and to update the probabilities of existing mappings.

1.8 Link Reweighting

The basic graph reinforcement algorithm is prone to producing irrelevant mappings by using character sequences with many different possible mappings as a bridge. Vowels were the most obvious examples of such character sequences. For example, automatic alignment produced 26 Hindi character sequences that map to the English letter "a", most of which were erroneous such as the mapping between "a" and "व" (pronounced va). Graph reinforcement resulted in many more such mappings. After successive iterations, such character sequences would cause the graph to be fully connected and eventually the link weights will tend to be uniform in their values. To illustrate this effect, we experimented with basic graph reinforcement on the NEWS dataset. The figures of merit were precision, recall, and F1-measure. Figures 3, 4, 5, and 6 show reinforcement results for Arabic, Russian, Hindi, and Tamil respectively. The figures show that: recall increased quickly and nearly saturated after several iterations; precision continued to drop with more iterations; and F1-measure peaked after a few iterations and began to drop afterwards. This behavior was undesirable because overall F1-measure values did not converge with iterations, necessitating the need to find clear stopping conditions.

To avoid this effect and to improve precision, we applied link reweighting after each iteration. Link reweighting had the effect of decreasing the weights of target character sequences that have many source character sequences mapping to them and hence reducing the effect of incorrectly inducing mappings. Link reweighting was performed as follows (Eq. 2):

$$m'(s|t) = \frac{m(s|t)}{\sum_{s_i \in S} m(s_i|t)}$$

Where $s_i \in S$ is a source character sequence that maps to *t*. So in the case of "a" mapping to the "व" character in Hindi, the link weight from "a" to "व" is divided by the sum of link weights from "a" to all 26 characters to which "a" maps.

We performed multiple experiments on the NEWS dataset to test the effect of graph reinforcement with link reweighting with varying number of reinforcement iterations. Figures 7, 8, 9, and 10 compare baseline results with smoothing to results with graph reinforcement at different iterations.

As can be seen in the figures, the F1-measure values stabilized as we performed multiple graph reinforcement iterations. Except for Russian, the results across different languages behaved in a similar manner.

For Russian, graph reinforcement marginally affected TM F1-measure, as precision and recall

marginally changed. The net improvement was 1.1 basis points. English and Russian do not share the same alphabet, and the number of initial mappings was bigger compared to the other language pairs. Careful inspection of the English-Russian test set, with the help of a Russian speaker, suggests that:
1) the test set reference contained many false negatives;
2) Russian names often have multiple phonetic forms (or spellings) in Russian with a single standard transliteration in English. For example, the Russian name "Olga" is often written and pronounced as "Ola" and "Olga" in Russian; and
3) certain English phones do not exist in Russian, leading to inconsistent character mappings in Russian. For example, the English phone for "g", as in "George", does not exist in Russian.

For the other languages, graph reinforcement yielded steadily improving recall and consequently steadily improving F1-measure. Most improvements were achieved within the first 5 iterations, and improvements beyond 10 iterations were generally small (less than 0.5 basis points in F1-measure). After 15 iterations, the improvements in overall F1-measure above the baseline with smoothing were 19.3, 5.3, and 32.8 basis points for Arabic, Tamil, and Hindi respectively. The F1-measure values seemed to stabilize with successive iterations. The least improvements were observed for Hindi. This could be attributed to the fact that Hindi spelling is largely phonetic, making letters in words pronounceable in only one way. This fact makes transliteration between Hindi and English easier than Arabic and Tamil. In the case of Tamil, the phonetics of letters change depending on the position of letters in words. As for Arabic, multiple letters sequences in English can map to single letters in Arabic and vice versa. Also, Arabic has diacritics which are typically omitted, but commonly transliterate to English vowels. Thus, the greater the difference in phonetics between two languages and the greater the phonetic complexity of either, the more TM can gain from the proposed technique.

1.9 When Graph Reinforcement Worked

An example where reinforcement worked entails the English-Arabic transliteration pair (Seljuq, سلاجقه). In the baseline runs with 1,000 training examples, both were not mapped to each other because there were no mappings between the letter "q" and the Arabic letter sequence "قه" (pronounced as "qah"). The only mappings that were available for "q" were "كه" (pronounced as "kah"), "ق" (pronounced as "q"), and "ك" (pronounced as "k") with probabilities 54.0, 0.10, and 5452 respectively. Intuitively, the third mapping is more likely than the second. After 3 graph reinforcement iterations, the top 5 mappings for "q" were "ق" (pronounced as "q"), "قه" (pronounced as "qah"), "كه" (pronounced as "kah"), "ك" (pronounced as "k"), and "الق" (pronounced as "alq") with mapping probabilities of 0.22, 0.19, 0.15, 0.05, and 0.05 respectively. In this case, graph reinforcement was able to find the missing mapping and properly reorder the mappings. Performing 10 iterations with link reweighting for Arabic led to 17 false positives. Upon examining them, we found that: 9 were actually correct, but erroneously labeled as false in the test set; 6 were phonetically similar like "اسبانيا" (pronounced espanya) and "Spain" and "التكنولوجيا" (pronounced alteknologya) and "technology"; and the remaining 2 were actually wrong, which were "بيتشي" (pronounced beatchi) and "medici" and "سيدي" (pronounced sidi) and "taya". This seems to indicate that graph reinforcement generally introduced more proper mappings than improper ones.

1.10 Comparing to the State-of-the-Art

Table 2 compares the best reported results in ACL-2010 NEWS TM shared task for Arabic (Noeman and Madkour, 2010) and for the other languages (Jiampojamarn et al. 2010) and the results obtained by the proposed technique using 10 iterations, with link reweighting. The comparison shows that the proposed algorithm yielded better results than the best reported results in the literature by 2.6, 4.8, 0.8 and 4.1 F1-measure points in Arabic, Russian, Hindi and Tamil respectively. For Arabic, the improvement over the previously reported result was due to improvement in precision, while for the other languages the improvements were due to improvements in both recall and precision.

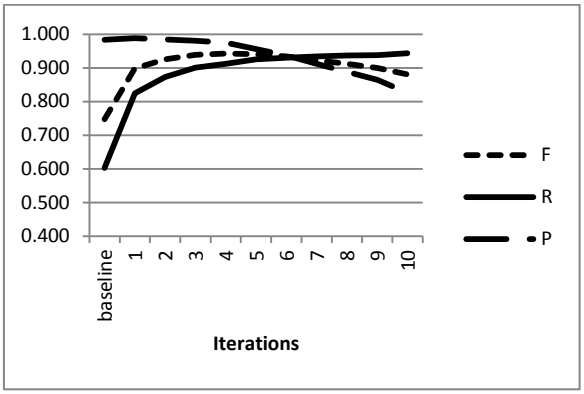


Figure 3: Graph reinforcement w/o link reweighting for Arabic

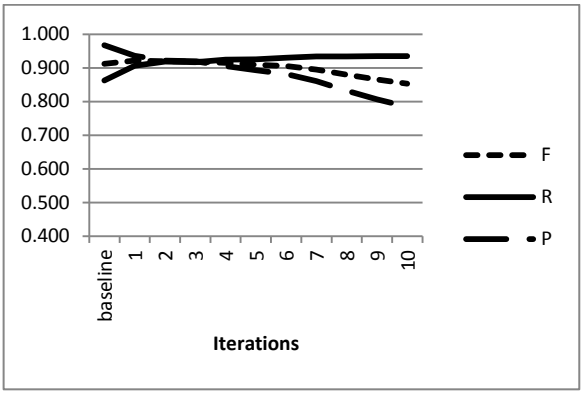


Figure 4: Graph reinforcement w/o link reweighting for Russian

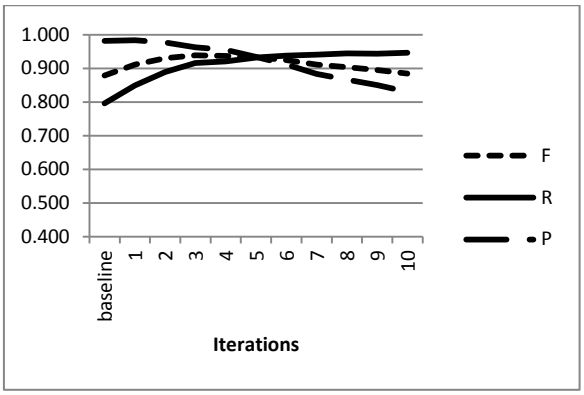


Figure 5: Graph reinforcement w/o link reweighting for Hindi

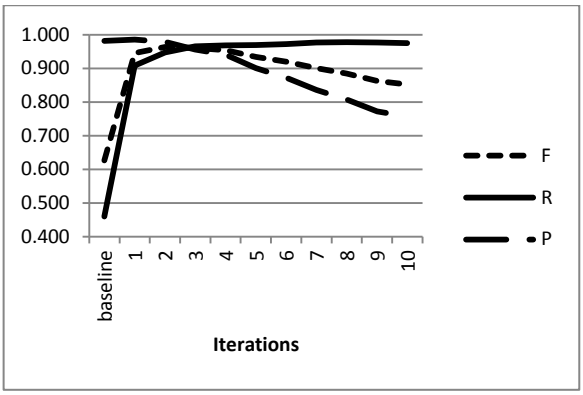


Figure 6: Graph reinforcement w/o link reweighting for Tamil

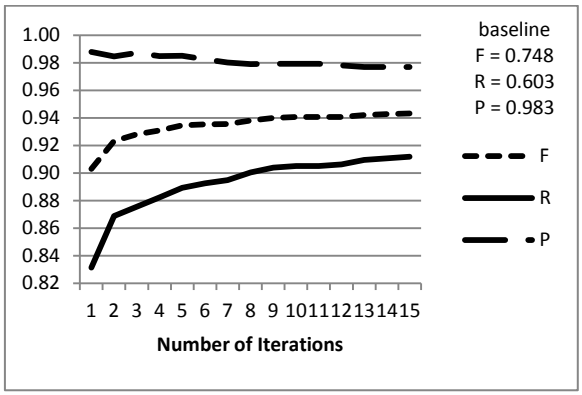


Figure 7: Graph reinforcement results for Arabic

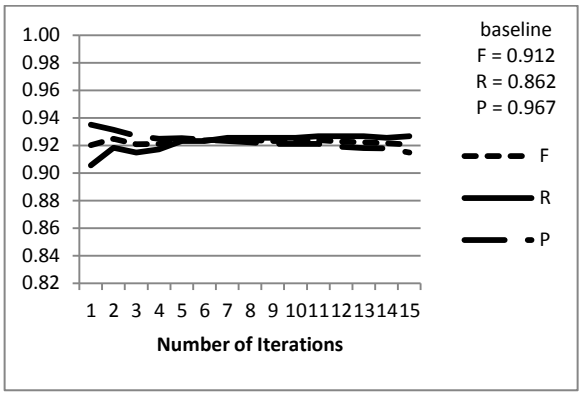


Figure 8: Graph reinforcement results for Russian

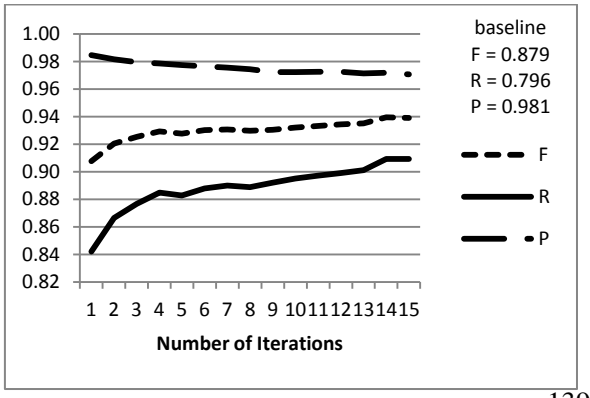


Figure 9: Graph reinforcement results for Hindi

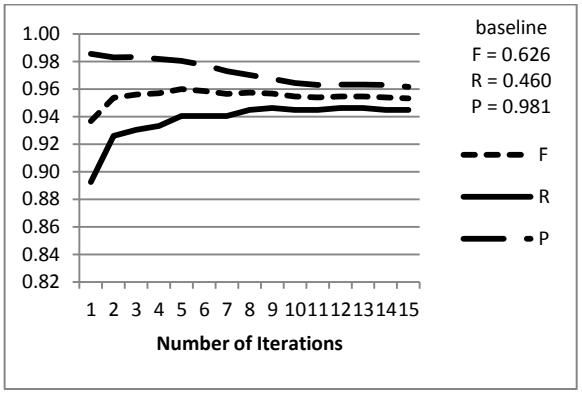


Figure 10: Graph reinforcement results for Tamil

	Shared Task			Proposed Algorithm		
English/	P	R	F	P	R	F
Arabic	0.887	0.945	0.915	0.979	0.905	0.941
Russian	0.880	0.869	0.875	0.921	0.925	0.923
Hindi	0.954	0.895	0.924	0.972	0.895	0.932
Tamil	0.923	0.906	0.914	0.964	0.945	0.955

Table 2: Best results obtained in ACL-2010 NEWS TM shared task compared to graph reinforcement with link reweighting after 10 iterations

Conclusion

In this paper, we presented a graph reinforcement algorithm with link reweighting to improve transliteration mining recall and precision by systematically inferring mappings that were unseen in training. We used the improved technique to extract transliteration pairs from parallel Wikipedia titles. The proposed technique solves two problems in transliteration mining, namely: some mappings may not be seen in training data – hurting recall, and certain mappings may not be seen a sufficient number of times to appropriate estimate mapping probabilities – hurting precision. The results showed that graph reinforcement yielded improved transliteration mining from parallel Wikipedia titles for all four languages on which the technique was tested.

Generally iterative graph reinforcement was able to induce unseen mappings in training data – improving recall. Link reweighting favored precision over recall counterbalancing the effect of graph reinforcement. The proposed system outperformed the best reported results in the literature for the ACL-2010 NEWS workshop shared task for Arabic, Russian, Hindi and Tamil. To extend the work, we would like to try transliteration mining from large comparable texts. The test parts of the NEWS dataset only contained short parallel fragments. For future work, graph reinforcement could be extended to MT to improve the coverage of aligned phrase tables. In doing so, it is reasonable to assume that there are multiple ways of expressing a singular concept and hence multiple translations are possible. Using graph reinforcement can help discover such translation though they may never be seen in training data. Using link reweighting in graph reinforcement can help demote unlikely translations while promoting likely ones. This could help clean MT phrase tables. Further, when dealing with transliteration, graph reinforcement can help find phonetic variations within a single language, which can have interesting applications in spelling correction and information retrieval. Applying the same to machine translation phrase tables can help identify paraphrases automatically.

References

Colin Bannard, Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. ACL-2005, pages 597—604.

Slaven Bilac, Hozumi Tanaka. 2005. Extracting transliteration pairs from comparable corpora. NLP-2005.

Eric Brill, Gary Kacmarcik, Chris Brockett. 2001. Automatically harvesting Katakana-English term pairs from search engine query logs. NLPRS 2001, pages 393–399.

Kareem Darwish. 2010. Transliteration Mining with Phonetic Conflation and Iterative Training. ACL NEWS workshop 2010.

Huang Fei, Stephan Vogel, and Alex Waibel. 2003. Extracting Named Entity Translingual Equivalence with Limited Resources. TALIP, 2(2):124–129.

Xiaodong He. 2007. Using Word-Dependent Transition Models in HMM based Word Alignment for Statistical Machine Translation. ACL-07 2nd SMT workshop.

Sittichai Jiampojamarn, Kenneth Dwyer, Shane Bergsma, Aditya Bhargava, Qing Dou, Mi-Young Kim and Grzegorz Kondrak. 2010. Transliteration Generation and Mining with Limited Training Resources. ACL NEWS workshop 2010.

Chengguo Jin, Dong-Il Kim, Seung-Hoon Na, Jong-Hyeok Lee. 2008. Automatic Extraction of English-Chinese Transliteration Pairs using Dynamic Window and Tokenizer. Sixth SIGHAN Workshop on Chinese Language Processing, 2008.

Alexandre Klementiev and Dan Roth. 2006. Named Entity Transliteration and Discovery from Multilingual Comparable Corpora. HLT Conf. of the North American Chapter of the ACL, pages 82–88.

Stanley Kok, Chris Brockett.. 2010. Hitting the Right Paraphrases in Good Time. Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, June 2010

A. Kumaran, Mitesh M. Khapra, Haizhou Li. 2010. Report of NEWS 2010 Transliteration Mining Shared Task. Proceedings of the 2010 Named Entities

Workshop, ACL 2010, pages 21–28, Uppsala, Sweden, 16 July 2010.

Jin-Shea Kuo, Haizhou Li, Ying-Kuei Yang. 2006. Learning Transliteration Lexicons from the Web. COLING-ACL2006, Sydney, Australia, 1129 – 1136.

Jin-shea Kuo, Haizhou Li, Ying-kuei Yang. 2007. A phonetic similarity model for automatic extraction of transliteration pairs. TALIP, 2007

Jin-Shea Kuo, Haizhou Li, Chih-Lung Lin. 2008. Mining Transliterations from Web Query Results: An Incremental Approach. Sixth SIGHAN Workshop on Chinese Language Processing, 2008.

Jin-shea Kuo, Ying-kuei Yang. 2005. Incorporating Pronunciation Variation into Extraction of Transliterated-term Pairs from Web Corpora. Journal of Chinese Language and Computing, 15 (1): (33-44).

Chun-Jen Lee, Jason S. Chang. 2003. Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. Workshop on Building and Using Parallel Texts, HLT-NAACL-2003, 2003.

Sara Noeman and Amgad Madkour. 2010. Language Independent Transliteration Mining System Using Finite State Automata Framework. ACL NEWS workshop 2010.

R. Mahesh, K. Sinha. 2009. Automated Mining Of Names Using Parallel Hindi-English Corpus. 7th Workshop on Asian Language Resources, ACL-IJCNLP 2009, pages 48–54, 2009.

Jong-Hoon Oh, Key-Sun Choi. 2006. Recognizing transliteration equivalents for enriching domain specific thesauri. 3rd Intl. WordNet Conf. (GWC-06), pages 231–237, 2006.

Jong-Hoon Oh, Hitoshi Isahara. 2006. Mining the Web for Transliteration Lexicons: Joint-Validation Approach. pp.254-261, 2006 IEEE/WIC/ACM Intl. Conf. on Web Intelligence (WI'06), 2006.

Yan Qu, Gregory Grefenstette, David A. Evans. 2003. Automatic transliteration for Japanese-to-English text retrieval. SIGIR 2003:353-360

Robert Russell. 1918. Specifications of Letters. US patent number 1,261,167.

K Saravanan, A Kumaran. 2008. Some Experiments in Mining Named Entity Transliteration Pairs from Comparable Corpora. The 2nd Intl. Workshop on Cross Lingual Information Access: Addressing the Need of Multilingual Societies, 2008.

Raghavendra Udupa, K. Saravanan, Anton Bakalov, Abhijit Bhole. 2009a. "They Are Out There, If You Know Where to Look": Mining Transliterations of OOV Query Terms for Cross-Language Information Retrieval. ECIR-2009, Toulouse, France, 2009.

Raghavendra Udupa, K. Saravanan, A. Kumaran, and Jagadeesh Jagarlamudi. 2009b. MINT: A Method for Effective and Scalable Mining of Named Entity Transliterations from Large Comparable Corpora. EACL 2009.

Raghavendra Udupa and Mitesh Khapra. 2010a. Transliteration Equivalence using Canonical Correlation Analysis. ECIR-2010, 2010.

Raghavendra Udupa, Shaishav Kumar. 2010b. Hashing-based Approaches to Spelling Correction of Personal Names. EMNLP 2010.

Gae-won You, Seung-won Hwang, Young-In Song, Long Jiang, Zaiqing Nie. 2010. Mining Name Translations from Entity Graph Mapping. Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 430–439.

Shiqi Zhao, Haifeng Wang, Ting Liu, Sheng Li. 2008. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. Proceedings of ACL-08: HLT, pages 780–788.

Experimental Support for a Categorical Compositional Distributional Model of Meaning

Edward Grefenstette
University of Oxford
Department of Computer Science
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
edward.grefenstette@cs.ox.ac.uk

Mehrnoosh Sadrzadeh
University of Oxford
Department of Computer Science
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
mehrs@cs.ox.ac.uk

Abstract

Modelling compositional meaning for sentences using empirical distributional methods has been a challenge for computational linguists. We implement the abstract categorical model of Coecke et al. (2010) using data from the BNC and evaluate it. The implementation is based on unsupervised learning of matrices for relational words and applying them to the vectors of their arguments. The evaluation is based on the word disambiguation task developed by Mitchell and Lapata (2008) for intransitive sentences, and on a similar new experiment designed for transitive sentences. Our model matches the results of its competitors in the first experiment, and betters them in the second. The general improvement in results with increase in syntactic complexity showcases the compositional power of our model.

1 Introduction

As competent language speakers, we humans can almost trivially make sense of sentences we've never seen or heard before. We are naturally good at understanding ambiguous words given a context, and forming the meaning of a sentence from the meaning of its parts. But while human beings seem comfortable doing this, machines fail to deliver. Search engines such as Google either fall back on bag of words models—ignoring syntax and lexical relations—or exploit superficial models of lexical semantics to retrieve pages with terms related to those in the query (Manning et al., 2008).

However, such models fail to shine when it comes to processing the semantics of phrases and sentences. Discovering the process of meaning assignment in natural language is among the most challenging and foundational questions of linguistics and computer science. The findings thereof will increase our understanding of cognition and intelligence and shall assist in applications to automating language-related tasks such as document search.

Compositional type-logical approaches (Montague, 1974; Lambek, 2008) and distributional models of lexical semantics (Schutze, 1998; Firth, 1957) have provided two partial orthogonal solutions to the question. Compositional formal semantic models stem from classical ideas from mathematical logic, mainly Frege's principle that the meaning of a sentence is a function of the meaning of its parts (Frege, 1892). Distributional models are more recent and can be related to Wittgenstein's later philosophy of 'meaning is use', whereby meanings of words can be determined from their context (Wittgenstein, 1953). The logical models relate to well known and robust logical formalisms, hence offering a scalable theory of meaning which can be used to reason inferentially. The distributional models have found their way into real world applications such as thesaurus extraction (Grefenstette, 1994; Curran, 2004) or automated essay marking (Landauer, 1997), and have connections to semantically motivated information retrieval (Manning et al., 2008). This two-sortedness of defining properties of meaning: 'logical form' versus 'contextual use', has left the quest for 'what is the foundational structure of meaning?' even more of a challenge.

Recently, Coecke et al. (2010) used high level cross-disciplinary techniques from logic, category

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1394–1404,
Edinburgh, Scotland, UK, July 27–31, 2011.

theory, and physics to bring the above two approaches together. They developed a unified mathematical framework whereby a sentence vector is by definition a function of the Kronecker product of its word vectors. A concrete instantiation of this theory was exemplified on a toy hand crafted corpus by Grefenstette et al. (2011). In this paper we implement it by training the model over the entire BNC. The highlight of our implementation is that words with relational types, such as verbs, adjectives, and adverbs are matrices that *act* on their arguments. We provide a general algorithm for building (or indeed learning) these matrices from the corpus.

The implementation is evaluated against the task provided by Mitchell and Lapata (2008) for disambiguating intransitive verbs, as well as a similar new experiment for transitive verbs. Our model improves on the best method evaluated in Mitchell and Lapata (2008) and offers promising results for the transitive case, demonstrating its scalability in comparison to that of other models. But we still feel there is need for a different class of experiments to showcase merits of compositionality in a statistically significant manner. Our work shows that the categorical compositional distributional model of meaning permits a practical implementation and that this opens the way to the production of large scale compositional models.

2 Two Orthogonal Semantic Models

Formal Semantics To compute the meaning of a sentence consisting of n words, meanings of these words must *interact* with one another. In formal semantics, this further interaction is represented as a function derived from the grammatical structure of the sentence, but meanings of words are amorphous objects of the domain: no distinction is made between words that have the same type. Such models consist of a pairing of syntactic interpretation rules (in the form of a grammar) with semantic interpretation rules, as exemplified by the simple model presented in Figure 1.

The parse of a sentence such as "cats like milk" typically produces its semantic interpretation by substituting semantic representation for their grammatical constituents and applying β-reduction where needed. Such a derivation is shown in Figure 2.

Syntactic Analysis	Semantic Interpretation
S $\rightarrow$ NP VP	$\|VP\|(\|NP\|)$
NP $\rightarrow$ cats, milk, etc.	$\|\text{cats}\|$, $\|\text{milk}\|$, ...
VP $\rightarrow$ Vt NP	$\|Vt\|(\|NP\|)$
Vt $\rightarrow$ like, hug, etc.	$\lambda yx.\|\text{like}\|(x, y)$, ...

Figure 1: A simple model of formal semantics.

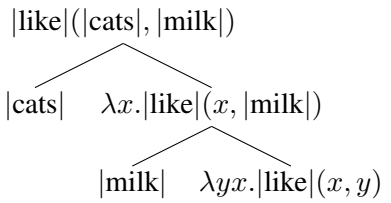


Figure 2: A parse tree showing a semantic derivation.

This methodology is used to translate sentences of natural language into logical formulae, then use computer-aided automation tools to reason about them (Alshawi, 1992). One major drawback is that the result of such analysis can only deal with truth or falsity as the meaning of a sentence, and says nothing about the closeness in meaning or topic of expressions beyond their truth-conditions and what models satisfy them, hence do not perform well on language tasks such as search. Furthermore, an underlying domain of objects and a valuation function must be provided, as with any logic, leaving open the question of how we might *learn* the meaning of language using such a model, rather than just use it.

Distributional Models Distributional models of semantics, on the other hand, dismiss the interaction between syntactically linked words and are solely concerned with lexical semantics. Word meaning is obtained empirically by examining the *contexts*[1] in which a word appears, and equating the meaning of a word with the distribution of contexts it shares. The intuition is that context of use is what we appeal to in learning the meaning of a word, and that words that frequently have the same sort of context in common are likely to be semantically related.

For instance, beer and sherry are both drinks, alcoholic, and often cause a hangover. We expect these facts to be reflected in a sufficiently large corpus: the words 'beer' and 'sherry' occur within the

[1] *E.g.* words which appear in the same sentence or n-word window, or words which hold particular grammatical or dependency relations to the word being learned.

context of identifying words such as 'drink', 'alcoholic' and 'hangover' more frequently than they occur with other content words.

Such context distributions can be encoded as vectors in a high dimensional space with contexts as basis vectors. For any word vector $\overrightarrow{\text{word}}$, the scalar weight c_i^{word} associated with each context basis vector $\overrightarrow{n_i}$ is a function of the number of times the word has appeared in that context. Semantic vectors $(c_1^{word}, c_2^{word}, \cdots, c_n^{word})$ are also denoted by sums of such weight/basis vector pairs:

$$\overrightarrow{\text{word}} = \sum_i c_i^{word} \overrightarrow{n_i}$$

Learning a semantic vector is just learning its basis weights from the corpus. This setting offers geometric means to reason about semantic similarity (*e.g.* via cosine measure or k-means clustering), as discussed in Widdows (2005).

The principal drawback of such models is their non-compositional nature: they ignore grammatical structure and logical words, and hence cannot compute the meanings of phrases and sentences in the same efficient way that they do for words. Common operations discussed in (Mitchell and Lapata, 2008) such as vector addition ($+$) and componentwise multiplication ($\odot$, *cf.* §4 for details) are commutative, hence if $\overrightarrow{vw} = \overrightarrow{v} + \overrightarrow{w}$ or $\overrightarrow{v} \odot \overrightarrow{w}$, then $\overrightarrow{vw} = \overrightarrow{wv}$, leading to unwelcome equalities such as

$$\overrightarrow{\text{the dog bit the man}} = \overrightarrow{\text{the man bit the dog}}$$

Non-commutative operations, such as the Kronecker product (*cf.* §4 for definition) can take word-order into account (Smolensky, 1990) or even some more complex syntactic relations, as described in Clark and Pulman (2007). However, the dimensionality of sentence vectors produced in this manner differs for sentences of different length, barring all sentences from being compared in the same vector space, and growing exponentially with sentence length hence quickly becoming computationally intractable.

3 A Hybrid Logico-Distributional Model

Whereas semantic compositional mechanisms for set-theoretic constructions are well understood, there are no obvious corresponding methods for vector spaces. To solve this problem, Coecke et al. (2010) use the abstract setting of category theory to turn the grammatical structure of a sentence into a morphism compatible with the higher level logical structure of vector spaces.

One pragmatic consequence of this abstract idea is as follows. In distributional models, there is a meaning vector for each word, *e.g.* $\overrightarrow{\text{cats}}$, $\overrightarrow{\text{like}}$, and $\overrightarrow{\text{milk}}$. The logical recipe tells us to *apply* the meaning of the verb to the meanings of subject and object. But how can a vector *apply* to other vectors? The solution proposed above implies that one needs to have different levels of meaning for words with different types. This is similar to logical models where verbs are relations and nouns are atomic sets. So verb vectors should be built differently from noun vectors, for instance as matrices.

The general information as to which words should be matrices and which words atomic vectors is in fact encoded in the type-logical representation of the grammatical structure of the sentence. This is the linear map with word vectors as input and sentence vectors as output. Hence, at least theoretically, one should be able to build sentence vectors and compare their synonymity in exactly the same way as one measures word synonymity.

Pregroup Grammars The aforementioned linear maps turn out to be the grammatical reductions of a type-logic called a Lambek pregroup grammar (Lambek, 2008)[2]. Pregroups and vector spaces share the same high level mathematical structure, referred to as a *compact closed category*, for a proof and details of this claim see Coecke et al. (2010); for a friendly introduction to category theory, see Coecke and Paquette (2011). One consequence of this parity is that the grammatical reductions of a pregroup grammar can be directly transformed into linear maps that act on vectors.

In a nutshell, pregroup types are either atomic or compound. Atomic types can be simple (*e.g.* n for noun phrases, s for statements) or left/right superscripted—referred to as adjoint types (*e.g.* n^r and n^l). An example of a compound type is that of a verb $n^r s n^l$. The superscripted types express that the verb is a relation with two arguments of type n,

[2]The usage of pregroup types is not essential, the types of any other logic, for instance CCG can be used, but should be translated into the language of pregroups.

which have to occur to the *right* and to the *left* of it, and that it outputs an argument of the type s. A transitive sentence has types as shown in Figure 3.

Each type n cancels out with its right adjoint n^r from the right and its left adjoint n^l from the left; mathematically speaking these mean[3]

$$n^l n \leq 1 \qquad \text{and} \qquad n n^r \leq 1$$

Here 1 is the unit of concatenation: $1n = n1 = n$. The corresponding grammatical reduction of a transitive sentence is $nn^r sn^l \leq 1s1 = s$. Each such reduction can be depicted as a wire diagram. The diagram of a transitive sentence is shown in Figure 3.

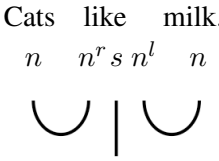


Figure 3: The pregroup types and reduction diagram for a transitive sentence.

Syntax-guided Semantic Composition According to Coecke et al. (2010) and based on a general completeness theorem between compact categories, wire diagrams, and vector spaces, the meaning of sentences can be canonically reduced to linear algebraic formulae. The following is the meaning vector of our transitive sentence:

$$\overrightarrow{\text{cats like milk}} = (f)\Big(\overrightarrow{\text{cats}} \otimes \overrightarrow{\text{like}} \otimes \overrightarrow{\text{milk}}\Big) \qquad (\mathbf{I})$$

Here f is the linear map that encodes the grammatical structure. The categorical morphism corresponding to it is denoted by the tensor product of 3 components: $\epsilon_V \otimes 1_S \otimes \epsilon_W$, where V and W are subject and object spaces, S is the sentence space, the ϵ's are the cups, and 1_S is the straight line in the diagram. The cups stand for taking inner products, which when done with the basis vectors imitate substitution. The straight line stands for the identity map that does nothing. By the rules of the category, equation **(I)** reduces to the following linear algebraic formula with lower dimensions, hence the dimensional explosion problem for Kronecker products is avoided:

$$\sum_{itj} c_{itj} \langle \overrightarrow{\text{cats}} | \overrightarrow{v_i} \rangle \, \overrightarrow{s_t} \, \langle \overrightarrow{w_j} | \overrightarrow{\text{milk}} \rangle \in S \qquad (\mathbf{II})$$

$\overrightarrow{v_i}, \overrightarrow{w_j}$ are basis vectors of V and W. The inner product $\langle \overrightarrow{\text{cats}} \mid \overrightarrow{v_i} \rangle$ substitutes the weights of $\overrightarrow{\text{cats}}$ into the first argument place of the verb (similarly for object and second argument place). $\overrightarrow{s_t}$ is a basis vector of the sentence space S in which meanings of sentences live, regardless of their grammatical structure.

The degree of synonymity of sentences is obtained by taking the cosine measure of their vectors. S is an abstract space: it needs to be instantiated to provide concrete meanings and synonymity measures. For instance, a truth-theoretic model is obtained by taking the sentence space S to be the 2-dimensional space with basis vectors $|1\rangle$ (True) and $|0\rangle$ (False).

4 Building Matrices for Relational Words

In this section we present a general scheme to build matrices for relational words. Recall that given a vector space A with basis $\{\overrightarrow{n_i}\}_i$, the Kronecker product of two vectors $\overrightarrow{v} = \sum_i c_i^a \overrightarrow{n_i}$ and $\overrightarrow{w} = \sum_i c_i^b \overrightarrow{n_i}$ is defined as follows:

$$\overrightarrow{v} \otimes \overrightarrow{w} = \sum_{ij} c_i^a c_j^b \, (\overrightarrow{n_i} \otimes \overrightarrow{n_j})$$

where $(\overrightarrow{n_i} \otimes \overrightarrow{n_j})$ is just the pairing of the basis of A, *i.e.* $(\overrightarrow{n_i}, \overrightarrow{n_j})$. The Kronecker product vectors belong in the tensor product of A with itself: $A \otimes A$, hence if A has dimension r, these will be of dimensionality $r \times r$. The point-wise multiplication of these vectors is defined as follows

$$\overrightarrow{v} \odot \overrightarrow{w} = \sum_i c_i^a c_i^b \, \overrightarrow{n_i}$$

The intuition behind having a matrix for a relational word is that any relation R on sets X and Y, *i.e.* $R \subseteq X \times Y$ can be represented as a matrix, namely one that has as row-bases $x \in X$ and as column-bases $y \in Y$, with weight $c_{xy} = 1$ where $(x, y) \in R$ and 0 otherwise. In a distributional setting, the weights, which are natural or real numbers,

[3] The relation $\leq$ is the partial order of the pregroup. It corresponds to implication $\Longrightarrow$ in a logical reading thereof. If these inequalities are replaced by equalities, *i.e.* if $n^l n = 1 = n n^r$, then the pregroup collapses into a group where $n^l = n^r$.

will represent more: 'the extent according to which x and y are related'. This can be determined in different ways.

Suppose X is the set of animals, and 'chase' is a relation on it: $chase \subseteq X \times X$. Take $x =$ 'dog' and $y =$ 'cat': with our type-logical glasses on, the obvious choice would be to take c_{xy} to be the number of times 'dog' has chased 'cat', *i.e.* the number of times the sentence 'the dog chases the cat' has appeared in the corpus. But in the distributional setting, this method will be too syntactic and dismissive of the actual *meaning* of 'cat' and 'dog'. If instead the corpus contains the sentence 'the hound hunted the wild cat', c_{xy} will be 0, restricting us to only assign meaning to sentences that have directly appeared in the corpus. We propose to, instead, use a level of abstraction by taking words such as verbs to be distributions over the semantic information in the vectors of their context words, rather than over the context words themselves.

Start with an r-dimensional vector space N with basis $\{\overrightarrow{n}_i\}_i$, in which meaning vectors of atomic words, such as nouns, live. The basis vectors of N are in principle all the words from the corpus, however in practice and following Mitchell and Lapata (2008) we had to restrict these to a subset of the most occurring words. These basis vectors are not restricted to nouns: they can as well be verbs, adjectives, and adverbs, so that we can define the meaning of a noun in all possible contexts—as is usual in context-based models—and not only in the context of other nouns. Note that basis words with relational types are treated as pure lexical items rather than as semantic objects represented as matrices. In short, we count how many times a noun has occurred close to words of other syntactic types such as 'elect' and 'scientific', rather than count how many times it has occurred close to their corresponding matrices: it is the lexical tokens that form the context, not their meaning.

Each relational word P with grammatical type π and m adjoint types $\alpha_1, \alpha_2, \cdots, \alpha_m$ is encoded as an $(r \times \ldots \times r)$ matrix with m dimensions. Since our vector space N has a fixed basis, each such matrix is represented in vector form as follows:

$$\overrightarrow{\mathrm{P}} = \sum_{\underbrace{ij\cdots\zeta}_{m}} c_{ij\cdots\zeta} \underbrace{(\overrightarrow{n}_i \otimes \overrightarrow{n}_j \otimes \cdots \otimes \overrightarrow{n}_\zeta)}_{m}$$

This vector lives in the tensor space $\underbrace{N \otimes N \otimes \cdots \otimes N}_{m}$. Each $c_{ij\cdots\zeta}$ is computed according to the procedure described in Figure 4.

1) Consider a sequence of words containing a relational word 'P' and its arguments w_1, w_2, $\cdots$, w_m, occurring in the same order as described in P's grammatical type π. Refer to these sequences as 'P'-relations. Suppose there are k of them.
2) Retrieve the vector $\overrightarrow{\mathrm{w}}_l$ of each argument w_l.
3) Suppose w_1 has weight c_i^1 on basis vector $\overrightarrow{n}_i$, w_2 has weight c_j^2 on basis vector $\overrightarrow{n}_j$, $\cdots$, and w_m has weight c_ζ^m on basis vector $\overrightarrow{n}_\zeta$. Multiply these weights

$$c_i^1 \times c_j^2 \times \cdots \times c_\zeta^m$$

4) Repeat the above steps for all the k 'P'-relations, and sum[a] the corresponding weights

$$c_{ij\cdots\zeta} = \sum_k \left(c_i^1 \times c_j^2 \times \cdots \times c_\zeta^m\right)_k$$

[a] We also experimented with multiplication, but the sparsity of noun vectors resulted in most verb matrices being empty.

Figure 4: Procedure for learning weights for matrices of words 'P' with relational types π of m arguments.

Linear algebraically, this procedure corresponds to computing the following

$$\overrightarrow{\mathrm{P}} = \sum_k \left(\overrightarrow{\mathrm{w}}_1 \otimes \overrightarrow{\mathrm{w}}_2 \otimes \cdots \otimes \overrightarrow{\mathrm{w}}_m\right)_k$$

Type-logical examples of relational words are verbs, adjectives, and adverbs. A transitive verb is represented as a 2 dimensional matrix since its type is $n^r s n^l$ with two adjoint types n^r and n^l. The corresponding vector of this matrix is

$$\overrightarrow{\mathrm{verb}} = \sum_{ij} c_{ij} \left(\overrightarrow{n}_i \otimes \overrightarrow{n}_j\right)$$

The weight c_{ij} corresponding to basis vector $\overrightarrow{n}_i \otimes \overrightarrow{n}_j$, is the extent according to which words that have co-occurred with $\overrightarrow{n}_i$ have been the subject of the 'verb' and words that have co-occurred with $\overrightarrow{n}_j$ have been the object of the 'verb'. This example computation is demonstrated in Figure 5.

1) Consider phrases containing 'verb', its subject w_1 and object w_2. Suppose there are k of them.
2) Retrieve vectors $\overrightarrow{\mathrm{w}}_1$ and $\overrightarrow{\mathrm{w}}_2$.
3) Suppose $\overrightarrow{\mathrm{w}}_1$ has weight c_i^1 on $\overrightarrow{n}_i$ and $\overrightarrow{\mathrm{w}}_2$ has c_j^2 on $\overrightarrow{n}_j$. Multiply these weights $c_i^1 \times c_j^2$.
4) Repeat the above steps for all k 'verb'-relations and sum the corresponding weights $\sum_k (c_i^1 \times c_j^2)_k$

Figure 5: Procedure for learning weights for matrices of transitive verbs.

Linear algebraically, we are computing

$$\overrightarrow{\text{verb}} \quad = \quad \sum_k \left(\overrightarrow{\mathrm{w}}_1 \otimes \overrightarrow{\mathrm{w}}_2\right)_k$$

As an example, consider the verb 'show' and suppose there are two 'show'-relations in the corpus:

s_1 = table show result
s_2 = map show location

The vector of 'show' is

$$\overrightarrow{\text{show}} = \overrightarrow{\text{table}} \otimes \overrightarrow{\text{result}} + \overrightarrow{\text{map}} \otimes \overrightarrow{\text{location}}$$

Consider an N space with four basis vectors 'far', 'room', 'scientific', and 'elect'. The TF/IDF-weighted values for vectors of the above four nouns (built from the BNC) are as shown in Table 1.

i	$\overrightarrow{\mathbf{n_i}}$	table	map	result	location
1	far	6.6	5.6	7	5.9
2	room	27	7.4	0.99	7.3
3	scientific	0	5.4	13	6.1
4	elect	0	0	4.2	0

Table 1: Sample weights for selected noun vectors.

Part of the matrix of 'show' is presented in Table 2.

As a sample computation, the weight c_{11} for vector $(1,1)$, *i.e.* $(\overrightarrow{\text{far}}, \overrightarrow{\text{far}})$ is computed by multiplying weights of 'table' and 'result' on $\overrightarrow{\text{far}}$, *i.e.* 6.6×7,

	far	room	scientific	elect
far	79.24	47.41	119.96	27.72
room	232.66	80.75	396.14	113.2
scientific	32.94	31.86	32.94	0
elect	0	0	0	0

Table 2: Sample semantic matrix for 'show'.

multiplying weights of 'map' and 'location' on $\overrightarrow{\text{far}}$, *i.e.* 5.6×5.9 then adding these $46.2 + 33.04$ and obtaining the total weight 79.24.

The same method is applied to build matrices for ditransitive verbs, which will have 3 dimensions, and adjectives and adverbs, which will be of 1 dimension each.

5 Computing Sentence Vectors

Meaning of sentences are vectors computed by taking the variables of the categorical prescription of meaning (the linear map f obtained from the grammatical reduction of the sentence) to be determined by the matrices of the relational words. For instance the meaning of the transitive sentence 'sub verb obj' is:

$$\overrightarrow{\text{sub verb obj}} = \sum_{itj} \langle \overrightarrow{\text{sub}} \mid \overrightarrow{v}_i \rangle \langle \overrightarrow{w}_j \mid \overrightarrow{\text{obj}} \rangle \, c_{itj} \, \overrightarrow{s}_t$$

We take $V := W := N$ and $S = N \otimes N$, then $\sum_{itj} c_{itj} \overrightarrow{s}_t$ is determined by the matrix of the verb, *i.e.* substitute it by $\sum_{ij} c_{ij} (\overrightarrow{n}_i \otimes \overrightarrow{n}_j)$[4]. Hence $\overrightarrow{\text{sub verb obj}}$ becomes:

$$\sum_{ij} \langle \overrightarrow{\text{sub}} \mid \overrightarrow{n}_i \rangle \langle \overrightarrow{n}_j \mid \overrightarrow{\text{obj}} \rangle c_{ij} (\overrightarrow{n}_i \otimes \overrightarrow{n}_j) \quad = \quad \sum_{ij} c_i^{sub} c_j^{obj} c_{ij} (\overrightarrow{n}_i \otimes \overrightarrow{n}_j)$$

This can be decomposed to point-wise multiplication of two vectors as follows:

$$\left(\sum_{ij} c_i^{sub} c_j^{obj} (\overrightarrow{n}_i \otimes \overrightarrow{n}_j) \right) \odot \left(\sum_{ij} c_{ij} (\overrightarrow{n}_i \otimes \overrightarrow{n}_j) \right)$$

[4]Note that by doing so we are also reducing the verb space from $N \otimes (N \otimes N) \otimes N$ to $N \otimes N$, since for our construction we only need tuples of the form $\overrightarrow{n}_i \otimes \overrightarrow{n}_i \otimes \overrightarrow{n}_j \otimes \overrightarrow{n}_j$ which are isomorphic to pairs $(\overrightarrow{n}_i \otimes \overrightarrow{n}_j)$.

The left argument is the Kronecker product of subject and object vectors and the right argument is the vector of the verb, so we obtain

$$\left(\overrightarrow{\text{sub}} \otimes \overrightarrow{\text{obj}}\right) \odot \overrightarrow{\text{verb}}$$

Since $\odot$ is commutative, this provides us with a distributional version of the type-logical meaning of the sentence: point-wise multiplication of the meaning of the verb to the Kronecker product of its subject and object:

$$\overrightarrow{\text{sub verb obj}} \quad = \quad \overrightarrow{\text{verb}} \odot \left(\overrightarrow{\text{sub}} \otimes \overrightarrow{\text{obj}}\right)$$

This mathematical operation can be informally described as a structured 'mixing' of the information of the subject and object, followed by it being 'filtered' through the information of the verb applied to them, in order to produce the information of the sentence.

In the transitive case, $S = N \otimes N$, hence $\overrightarrow{s}_t = \overrightarrow{n}_i \otimes \overrightarrow{n}_j$. More generally, the vector space corresponding to the abstract sentence space S is the concrete tensor space $(N \otimes \ldots \otimes N)$ for m the dimension of the matrix of the 'verb'. As we have seen above, in practice we do not need to build this tensor space, as the computations thereof reduce to point-wise multiplications and summations.

Similar computations yield meanings of sentences with adjectives and adverbs. For instance the meaning of a transitive sentence with a modified subject and a modified verb we have

$$\overrightarrow{\text{adj sub verb obj adv}} = \left(\overrightarrow{\text{adv}} \odot \overrightarrow{\text{verb}}\right) \odot \left(\left(\overrightarrow{\text{adj}} \odot \overrightarrow{\text{sub}}\right) \otimes \overrightarrow{\text{obj}}\right)$$

After building vectors for sentences, we can compare their meaning and measure their degree of synonymy by taking their cosine measure.

6 Evaluation

Evaluating such a framework is no easy task. What to evaluate depends heavily on what sort of application a practical instantiation of the model is geared towards. In (Grefenstette et al., 2011), it is suggested that the simplified model we presented and expanded here could be evaluated in the same way as lexical semantic models, measuring compositionally built sentence vectors against a benchmark dataset such as that provided by Mitchell and Lapata (2008). In this section, we briefly describe the evaluation of our model against this dataset. Following this, we present a new evaluation task extending the experimental methodology of Mitchell and Lapata (2008) to transitive verb-centric sentences, and compare our model to those discussed by Mitchell and Lapata (2008) within this new experiment.

First Dataset Description The first experiment, described in detail by Mitchell and Lapata (2008), evaluates how well compositional models disambiguate ambiguous words given the context of a potentially disambiguating noun. Each entry of the dataset provides a noun, a target verb and landmark verb (both intransitive). The noun must be composed with both verbs to produce short phrase vectors the similarity of which is measured by the candidate. Also provided with each entry is a classification ("High" or "Low") indicating whether or not the verbs are indeed semantically close within the context of the noun, as well as an evaluator-set similarity score between 1 and 7 (along with an evaluator identifier), where 1 is low similarity and 7 is high.

Evaluation Methodology Candidate models provide a similarity score for each entry. The scores of high similarity entries and low similarity entries are averaged to produce a mean High score and mean Low score for the model. The correlation of the model's similarity judgements with the human judgements is also calculated using Spearman's ρ, a metric which is deemed to be more scrupulous, and ultimately that by which models should be ranked, by Mitchell and Lapata (2008). The mean for each model is on a $[0, 1]$ scale, except for UpperBound which is on the same $[1, 7]$ scale the annotators used. The ρ scores are on a $[-1, 1]$ scale. It is assumed that inter-annotator agreement provides the theoretical maximum ρ for any model for this experiment. The cosine measure of the verb vectors, ignoring the noun, is taken to be the baseline (no composition).

Other Models The other models we compare ours to are those evaluated by Mitchell and Lapata (2008). We provide a selection of the results

from that paper for the worst (Add) and best[5] (Multiply) performing models, as well as the previous second-best performing model (Kintsch). The additive and multiplicative models are simply applications of vector addition and component-wise multiplication. We invite the reader to consult (Mitchell and Lapata, 2008) for the description of Kintsch's additive model and parametric choices.

Model Parameters To provide the most accurate comparison with the existing multiplicative model, and exploiting the aforementioned feature that the categorical model can be built "on top of" existing lexical distributional models, we used the parameters described by Mitchell and Lapata (2008) to reproduce the vectors evaluated in the original experiment as our noun vectors. All vectors were built from a lemmatised version of the BNC. The noun basis was the 2000 most common context words, basis weights were the probability of context words given the target word divided by the overall probability of the context word. Intransitive verb function-vectors were trained using the procedure presented in §4. Since the dataset only contains intransitive verbs and nouns, we used $S = N$. The cosine measure of vectors was used as a similarity metric.

First Experiment Results In Table 3 we present the comparison of the selected models. Our categorical model performs significantly better than the existing second-place (Kintsch) and obtains a ρ quasi-identical to the multiplicative model, indicating significant correlation with the annotator scores.

There is not a large difference between the mean High score and mean Low score, but the distribution in Figure 6 shows that our model makes a non-negligible distinction between high similarity phrases and low similarity phrases, despite the absolute scores not being different by more than a few percentiles.

[5]The multiplicative model presented here is what is qualified as best in (Mitchell and Lapata, 2008). However, they also present a slightly better performing ($\rho = 0.19$) model which is a combination of their multiplicative model and a weighted additive model. The difference in ρ is qualified as "not statistically significant" in the original paper, and furthermore the mixed model requires parametric optimisation hence was not evaluated against the entire test set. For these reasons, we chose not to include it in the comparison.

Model	High	Low	ρ
Baseline	0.27	0.26	0.08
Add	0.59	0.59	0.04
Kintsch	0.47	0.45	0.09
Multiply	0.42	0.28	0.17
Categorical	**0.84**	**0.79**	**0.17**
UpperBound	4.94	3.25	0.40

Table 3: Selected model means for High and Low similarity items and correlation coefficients with human judgements, first experiment (Mitchell and Lapata, 2008). $p < 0.05$ for each ρ.

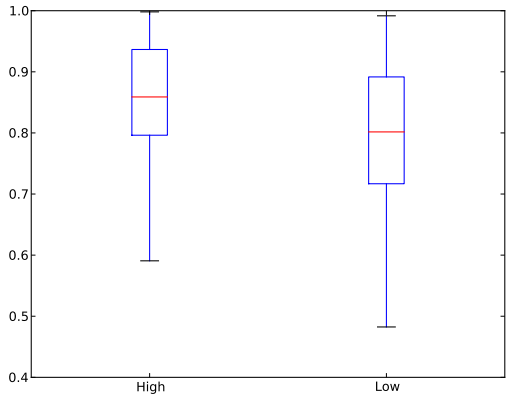


Figure 6: Distribution of predicted similarities for the categorical distributional model on High and Low similarity items.

Second Dataset Description The second dataset[6], developed by the authors, follows the format of the (Mitchell and Lapata, 2008) dataset used for the first experiment, with the exception that the target and landmark verbs are transitive, and an object noun is provided in addition to the subject noun, hence forming a small transitive sentence. The dataset comprises 200 entries consisting of sentence pairs (hence a total of 400 sentences) constructed by following the procedure outlined in §4 of (Mitchell and Lapata, 2008), using transitive verbs from CELEX[7]. For examples of these sentences, see Table 4. The dataset was split into four sections of 100 entries each, with guaranteed 50% exclusive overlap with

[6]http://www.cs.ox.ac.uk/activities/CompDistMeaning/GS2011data.txt

[7]http://celex.mpi.nl/

exactly two other datasets. Each section was given to a group of evaluators, with a total of 25, who were asked to form simple transitive sentence pairs from the verbs, subject and object provided in each entry; for instance 'the table showed the result' from 'table show result'. The evaluators were then asked to rate the semantic similarity of each verb pair within the context of those sentences, and offer a score between 1 and 7 for each entry. Each entry was given an arbitrary classification of HIGH or LOW by the authors, for the purpose of calculating mean high/low scores for each model. For example, the first two pairs in table 4 were classified as HIGH, whereas the second two pairs as LOW.

Sentence 1	Sentence 2
table show result	table express result
map show location	map picture location
table show result	table picture result
map show location	map express location

Table 4: Example entries from the transitive dataset without annotator score, second experiment.

Evaluation Methodology The evaluation methodology for the second experiment was identical to that of the first, as are the scales for means and scores. Here also, Spearman's ρ is deemed a more rigorous way of determining how well a model tracks difference in meaning. This is both because of the imprecise nature of the classification of verb pairs as HIGH or LOW; and since the objective similarity scores produced by a model that distinguishes sentences of different meaning from those of similar meaning can be renormalised in practice. Therefore the delta between HIGH means and LOW mean cannot serve as a definite indication of the practical applicability (or lack thereof) of semantic models; the means are provided just to aid comparison with the results of the first experiment.

Model Parameters As in the first experiment, the lexical vectors from (Mitchell and Lapata, 2008) were used for the other models evaluated (additive, multiplicative and baseline)[8] and for the noun vectors of our categorical model. Transitive verb vectors were trained as described in §4 with $S = N \otimes N$.

Second Experiment Results The results for the models evaluated against the second dataset are presented in Table 5.

Model	High	Low	ρ
Baseline	0.47	0.44	0.16
Add	0.90	0.90	0.05
Multiply	0.67	0.59	0.17
Categorical	**0.73**	**0.72**	**0.21**
UpperBound	4.80	2.49	0.62

Table 5: Selected model means for High and Low similarity items and correlation coefficients with human judgements, second experiment. $p < 0.05$ for each ρ.

We observe a significant (according to $p < 0.0.5$) improvement in the alignment of our categorical model with the human judgements, from 0.17 to 0.21. The additive model continues to make little distinction between senses of the verb during composition, and the multiplicative model's alignment does not change, but becomes statistically indistinguishable from the non-compositional baseline model.

Once again we note that the high-low means are not very indicative of model performance, as the difference between high mean and the low mean of the categorical model is much smaller than that of the both the baseline model and multiplicative model, despite better alignment with annotator judgements.

7 Discussion

In this paper, we described an implementation of the categorical model of meaning (Coecke et al., 2010), which combines the formal logical and the empirical distributional frameworks into a unified semantic model. The implementation is based on building matrices for words with relational types (adjectives, verbs), and vectors for words with atomic types (nouns), based on data from the BNC. We then show how to apply verbs to their subject/object, in order to compute the meaning of intransitive and transitive sentences.

[8] Kintsch was not evaluated as it required optimising model parameters against a held-out segment of the test set, and we could not replicate the methodology of Mitchell and Lapata (2008) with full confidence.

Other work uses matrices to model meaning (Baroni and Zamparelli, 2010; Guevara, 2010), but only for adjective-noun phrases. Our approach easily applies to such compositions, as well as to sentences containing combinations of adjectives, nouns, verbs, and adverbs. The other key difference is that they learn their matrices in a top-down fashion, *i.e.* by regression from the composite adjective-noun context vectors, whereas our model is bottom-up: it learns sentence/phrase meaning compositionally from the vectors of the compartments of the composites. Finally, very similar functions, for example a verb with argument alternations such as 'break' in 'Y breaks' and 'X breaks Y', are not treated as unrelated. The matrix of the intransitive 'break' uses the corpus-observed information about the subject of break, including that of 'Y', similarly the matrix of the transitive 'break' uses information about its subject and object, including that of 'X' and 'Y'. We leave a thorough study of these phenomena, which fall under providing a modular representation of passive-active similarities, to future work.

We evaluated our model in two ways: first against the word disambiguation task of Mitchell and Lapata (2008) for intransitive verbs, and then against a similar new experiment for transitive verbs, which we developed.

Our findings in the first experiment show that the categorical method performs on par with the leading existing approaches. This should not surprise us given that the context is so small and our method becomes similar to the multiplicative model of Mitchell and Lapata (2008). However, our approach is sensitive to grammatical structure, leading us to develop a second experiment taking this into account and differentiating it from models with commutative composition operations.

The second experiment's results deliver the expected qualitative difference between models, with our categorical model outperforming the others and showing an increase in alignment with human judgements in correlation with the increase in sentence complexity. We use this second evaluation principally to show that there is a strong case for the development of more complex experiments measuring not only the disambiguating qualities of compositional models, but also their syntactic sensitivity, which is not directly measured in the existing experiments.

These results show that the high level categorical distributional model, uniting empirical data with logical form, can be implemented just like any other concrete model. Furthermore it shows better results in experiments involving higher syntactic complexity. This is just the tip of the iceberg: the mathematics underlying the implementation ensures that it uniformly scales to larger, more complicated sentences and enables it to compare synonymity of sentences that are of different grammatical structure.

8 Future Work

Treatment of function words such as 'that', 'who', as well as logical words such as quantifiers and conjunctives are left to future work. This will build alongside the general guidelines of Coecke et al. (2010) and concrete insights from the work of Widdows (2005). It is not yet entirely clear how existing set-theoretic approaches, for example that of discourse representation and generalised quantifiers, apply to our setting. Preliminary work on integration of the two has been presented by Preller (2007) and more recently also by Preller and Sadrzadeh (2009).

As mentioned by one of the reviewers, our pregroup approach to grammar flattens the sentence representation, in that the verb is applied to its subject and object at the same time; whereas in other approaches such as CCG, it is first applied to the object to produce a verb phrase, then applied to the subject to produce the sentence. The advantages and disadvantages of this method and comparisons with other systems, in particular CCG, constitutes ongoing work.

9 Acknowledgement

We wish to thank P. Blunsom, S. Clark, B. Coecke, S. Pulman, and the anonymous EMNLP reviewers for discussions and comments. Support from EPSRC grant EP/F042728/1 is gratefully acknowledged by M. Sadrzadeh.

References

H. Alshawi (ed). 1992. *The Core Language Engine*. MIT Press.

M. Baroni and R. Zamparelli. 2010. *Nouns are vectors, adjectives are matrices*. Proceedings of Conference

on Empirical Methods in Natural Language Processing (EMNLP).

S. Clark and S. Pulman. 2007. *Combining Symbolic and Distributional Models of Meaning*. Proceedings of AAAI Spring Symposium on Quantum Interaction. AAAI Press.

B. Coecke, and E. Paquette. 2011. *Categories for the Practicing Physicist. New Structures for Physics*, 167-271. B. Coecke (ed.). Lecture Notes in Physics **813**. Springer.

B. Coecke, M. Sadrzadeh and S. Clark. 2010. *Mathematical Foundations for Distributed Compositional Model of Meaning*. Lambek Festschrift. Linguistic Analysis **36**, 345–384. J. van Benthem, M. Moortgat and W. Buszkowski (eds.).

J. Curran. 2004. *From Distributional to Semantic Similarity*. PhD Thesis, University of Edinburgh.

K. Erk and S. Padó. 2004. *A Structured Vector Space Model for Word Meaning in Context*. Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP), 897–906.

G. Frege 1892. *Über Sinn und Bedeutung*. Zeitschrift für Philosophie und philosophische Kritik 100.

J. R. Firth. 1957. *A synopsis of linguistic theory 1930-1955*. Studies in Linguistic Analysis.

E. Grefenstette, M. Sadrzadeh, S. Clark, B. Coecke, S. Pulman. 2011. *Concrete Compositional Sentence Spaces for a Compositional Distributional Model of Meaning*. International Conference on Computational Semantics (IWCS'11). Oxford.

G. Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer.

E. Guevara. 2010. *A Regression Model of Adjective-Noun Compositionality in Distributional Semantics*. Proceedings of the ACL GEMS Workshop.

Z. S. Harris. 1966. *A Cycling Cancellation-Automaton for Sentence Well-Formedness*. International Computation Centre Bulletin **5**, 69–94.

R. Hudson. 1984. *Word Grammar*. Blackwell.

J. Lambek. 2008. *From Word to Sentence*. Polimetrica, Milan.

T. Landauer, and S. Dumais. 2008. *A solution to Platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge*. Psychological review.

C. D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

J. Mitchell and M. Lapata. 2008. *Vector-based models of semantic composition*. Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, 236–244.

R. Montague. 1974. *English as a formal language*. Formal Philosophy, 189–223.

J. Nivre. 2003. *An efficient algorithm for projective dependency parsing*. Proceedings of the 8th International Workshop on Parsing Technologies (IWPT).

A. Preller. *Towards Discourse Representation via Pregroup Grammars*. Journal of Logic Language Information **16** 173–194.

A. Preller and M. Sadrzadeh. *Semantic Vector Models and Functional Models for Pregroup Grammars*. Journal of Logic Language Information. DOI: 10.1007/s10849-011-9132-2. to appear.

J. Saffron, E. Newport, R. Asling. 1999. *Word Segmentation: The role of distributional cues*. Journal of Memory and Language **35**, 606–621.

H. Schuetze. 1998. *Automatic Word Sense Discrimination*. Computational Linguistics **24**, 97–123.

P. Smolensky. 1990. *Tensor product variable binding and the representation of symbolic structures in connectionist systems*. Computational Linguistics **46, 1–2**, 159–216.

M. Steedman. 2000. *The Syntactic Process*. MIT Press.

D. Widdows. 2005. *Geometry and Meaning*. University of Chicago Press.

L. Wittgenstein. 1953. *Philosophical Investigations*. Blackwell.

Cross-Cutting Models of Lexical Semantics

Joseph Reisinger
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
joeraii@cs.utexas.edu

Raymond Mooney
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
mooney@cs.utexas.edu

Abstract

Context-dependent word similarity can be measured over multiple *cross-cutting* dimensions. For example, *lung* and *breath* are similar thematically, while *authoritative* and *superficial* occur in similar syntactic contexts, but share little semantic similarity. Both of these notions of similarity play a role in determining word meaning, and hence lexical semantic models must take them both into account. Towards this end, we develop a novel model, Multi-View Mixture (MVM), that represents words as multiple *overlapping clusterings*. MVM finds multiple data partitions based on different subsets of features, subject to the marginal constraint that feature subsets are distributed according to Latent Dirichlet Allocation. Intuitively, this constraint favors feature partitions that have coherent topical semantics. Furthermore, MVM uses soft feature assignment, hence the contribution of each data point to each clustering view is variable, isolating the impact of data only to views where they assign the most features. Through a series of experiments, we demonstrate the utility of MVM as an inductive bias for capturing relations between words that are intuitive to humans, outperforming related models such as Latent Dirichlet Allocation.

1 Introduction

Humans categorize objects using multiple orthogonal taxonomic systems, where category generalization depends critically on what features are relevant to one particular system. For example, foods can be organized in terms of their nutritional value (high in fiber) or situationally (commonly eaten for Thanksgiving; Shafto et al. (2006)). Human knowledgebases such as Wikipedia also exhibit such multiple clustering structure (e.g. people are organized by occupation or by nationality). The effects of these overlapping categorization systems manifest themselves at the lexical semantic level (Murphy, 2002), implying that lexicographical word senses and traditional computational models of word-sense based on clustering or exemplar activation are too impoverished to capture the rich dynamics of word usage.

In this work, we introduce a novel probabilistic clustering method, Multi-View Mixture (MVM), based on *cross-cutting categorization* (Shafto et al., 2006) that generalizes traditional *vector-space* or *distributional* models of lexical semantics (Curran, 2004; Padó and Lapata, 2007; Schütze, 1998; Turney, 2006). Cross-cutting categorization finds multiple feature subsets (categorization systems) that produce high quality clusterings of the data. For example words might be clustered based on their part of speech, or based on their thematic usage. Context-dependent variation in word usage can be accounted for by leveraging multiple latent categorization systems. In particular, cross-cutting models can be used to capture both *syntagmatic* and *paradigmatic* notions of word relatedness, breaking up word features into multiple categorization systems and then computing similarity separately for each system.

MVM leverages primitives from Dirichlet-Process Mixture Models (DPMMs) and Latent Dirichlet Allocation (LDA). Each clustering (*view*) in MVM consists of a distribution over features and data and views are further subdivided into clusters based on a DPMM. View marginal distributions are determined by LDA, allowing data features to be distributed over multiple views, explaining subsets of features.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1405–1415,
Edinburgh, Scotland, UK, July 27–31, 2011.

We evaluate MVM against several other model-based clustering procedures in a series of human evaluation tasks, measuring its ability to find meaningful syntagmatic and paradigmatic structure. We find that MVM finds more semantically and syntactically coherent fine-grained structure, using both common and rare n-gram contexts.

2 Mixture Modeling and Lexical Semantics

Distributional, or *vector space* methods attempt to model word meaning by embedding words in a common metric space, whose dimensions are derived from, e.g., word collocations (Schütze, 1998), syntactic relations (Padó and Lapata, 2007), or latent semantic spaces (Finkelstein et al., 2001; Landauer and Dumais, 1997; Turian et al., 2010). The distributional hypothesis addresses the problem of modeling word similarity (Curran, 2004; Miller and Charles, 1991; Schütze, 1998; Turney, 2006), and can be extended to selectional preference (Resnik, 1997) and lexical substitution (McCarthy and Navigli, 2007) as well. Such methods are highly scalable (Gorman and Curran, 2006) and have been applied in information retrieval (Manning et al., 2008), large-scale taxonomy induction (Snow et al., 2006), and knowledge acquisition (Van Durme and Paşca, 2008).

Vector space models fail to capture the richness of word meaning since similarity is not a globally consistent metric. It violates, e.g., the triangle inequality: the sum of distances from *bat* to *club* and *club* to *association* is less than the distance from *bat* to *association* (Griffiths et al., 2007; Tversky and Gati, 1982).[1] Erk (2007) circumvents this problem by representing words as multiple exemplars derived directly from word occurrences and embedded in a common vector space to capture context-dependent usage. Likewise Reisinger and Mooney (2010) take a similar approach using mixture modeling combined with a background variation model to generate multiple prototype vectors for polysemous words.

Both of these approaches still ultimately embed all words in a single metric space and hence argue for globally consistent metrics that capture human intuitive notions of "similarity." Rather than assuming a global metric embedding exists, in this work we simply leverage the *cluster assumption*, e.g. that similar words should appear in the same clusters, in particular extending it to multiple clusterings. The cluster assumption is a natural fit for lexical semantics, as partitions can account for metric violations. The end result is a model capable of representing multiple, overlapping similarity metrics that result in disparate valid clusterings leveraging the

> **Subspace Hypothesis**: For any pair of words, the set of "active" features governing their apparent similarity differs. For example *wine* and *bottle* are similar and *wine* and *vinegar* are similar, but it would not be reasonable to expect that the features governing such similarity computations to overlap much, despite occurring in similar documents.

MVM can extract multiple competing notions of similarity, for example both *paradigmatic*, or thematic similarity, and *syntagmatic* or syntactic similarity, in addition to more fine grained relations.

3 Multi-View Clustering with MVM

As feature dimensionality increases, the number of ways the data can exhibit interesting structure goes up exponentially. Clustering is commonly used to explain data, but often there are several equally valid, competing clusterings, keying off of different subsets of features, especially in high-dimensional settings such as text mining (Niu et al., 2010). For example, company websites can be clustered by sector or by geographic location, with one particular clustering becoming predominant when a majority of features correlate with it. In fact, informative features in one clustering may be noise in another, e.g. the occurrence of *CEO* is not necessarily discriminative when clustering companies by industry sector, but may be useful in other clusterings. Multiple clustering is one approach to inferring feature subspaces that lead to high quality data partitions. Multiple clustering also improves the flexibility of generative clustering models, as a single model is no longer required to explain all the variance in the feature dimensions (Mansinghka et al., 2009).

[1] Similarity also has been shown to violate symmetry (e.g. people have the intuition that *China* is more similar to *North Korea* than *North Korea* is to *China*).

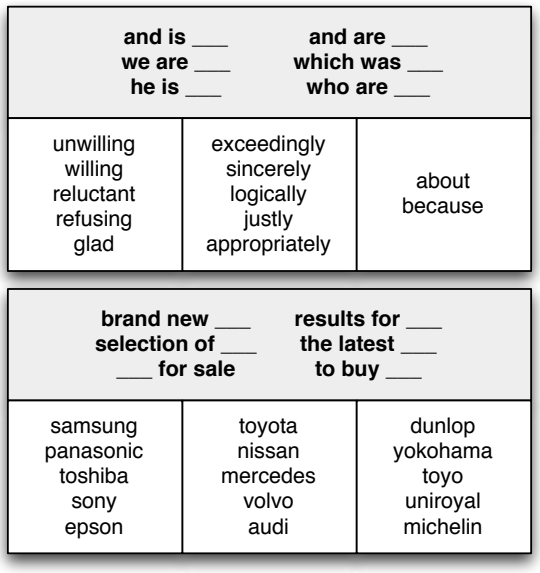


Figure 1: Example clusterings from MVM applied to Google n-gram data. Top contexts (features) for each view are shown, along with examples of word clusters. Although these particular examples are interpretable, in general the relationship captured by the view's context subspace is not easily summarized.

MVM is a multinomial-Dirichlet multiple clustering procedure for distributional lexical semantics that fits multiple, overlapping Dirichlet Process Mixture Models (DPMM) to a set of word data. Features are distributed across the set of clusterings (views) using LDA, and each DPMM is fit using a subset of the features. This reduces clustering noise and allows MVM to capture multiple ways in which the data can be partitioned. Figure 1 shows a simple example, and Figure 2 shows a larger sample of feature-view assignments from a 3-view MVM fit to contexts drawn from the Google n-gram corpus.

We implement MVM using generative model primitives drawn from Latent Dirichlet Allocation (LDA) and the Dirichlet Process (DP). $|M|$ disparate clusterings (views) are inferred jointly from a set of data $\mathcal{D} = \{\mathbf{w}_d | d \in [1 \dots D]\}$. Each data vector $\mathbf{w}_d$ is associated with a probability distribution over views $\boldsymbol{\theta}_d^{|M|}$. Empirically, $\boldsymbol{\theta}_d^{|M|}$ is represented as a set of *feature-view* assignments $\mathbf{z}_d$, sampled via the standard LDA collapsed Gibbs sampler. Hence, each view maintains a separate distribution over features. The generative model for feature-view assignment is given by

$$\begin{aligned}
\boldsymbol{\theta}_d^{|M|} | \boldsymbol{\alpha} &\sim \text{Dirichlet}(\boldsymbol{\alpha}), & d &\in D, \\
\boldsymbol{\phi}_m | \boldsymbol{\beta} &\sim \text{Dirichlet}(\boldsymbol{\beta}), & m &\in |M|, \\
z_{dn} | \boldsymbol{\theta}_d &\sim \text{Discrete}(\boldsymbol{\theta}_d), & n &\in |\mathbf{w}_d|, \\
w_{dn} | \boldsymbol{\phi}_{z_{dn} m} &\sim \text{Discrete}(\boldsymbol{\phi}_{z_{dn} m}), & n &\in |\mathbf{w}_d|,
\end{aligned}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are hyperparameters smoothing the per-document topic distributions and per-topic word distributions respectively.

Conditional on the feature-view assignment $\{\mathbf{z}\}$, a clustering is inferred for each view using the Chinese Restaurant Process representation of the DP. The clustering probability is given by

$$\begin{aligned}
p(\mathbf{c} | \mathbf{z}, \mathbf{w}) &\propto p(\{\mathbf{c}_m\}, \mathbf{z}, \mathbf{w}) \\
&= \prod_{m=1}^{M} \prod_{d=1}^{|D|} p(\mathbf{w}_d^{[\mathbf{z}=m]} | \mathbf{c}_m, \mathbf{z}) p(\mathbf{c}_m | \mathbf{z}).
\end{aligned}$$

where $p(\mathbf{c}_m | \mathbf{z})$ is a prior on the clustering for view m, i.e. the DPMM, and $p(\mathbf{w}_d^{[\mathbf{z}=m]} | \mathbf{c}_m, \mathbf{z})$ is the likelihood of the clustering $\mathbf{c}_m$ given the data point $\mathbf{w}_d$ restricted to the features assigned to view m:

$$\mathbf{w}_d^{[\mathbf{z}=m]} \stackrel{\text{def}}{=} \{w_{id} | z_{id} = m\}.$$

Thus, we treat the m clusterings $\mathbf{c}_m$ as conditionally independent given the feature-view assignments.

The feature-view assignments $\{\mathbf{z}\}$ act as a set of marginal constraints on the multiple clusterings, and the impact that each data point can have on each clustering is limited by the number of features assigned to it. For example, in a two-view model, $z_{id} = 1$ might be set for all syntactic features (yielding a syntagmatic clustering) while $z_{id} = 2$ is set for document features (paradigmatic clustering).

By allowing the clustering model capacity to vary via the DPMM, MVM can naturally account for the semantic variance of the view. This provides a novel mechanism for handling feature noise: noisy features can be assigned to a separate view with potentially a small number of clusters. This phenomenon is apparent in cluster 1, view 1 in the example in figure 2, where place names and adjectives are clustered together using rare contexts

From a topic modeling perspective, MVM finds topic refinements within each view, similar to hierarchical methods such as the nested Chinese Restaurant Process (Blei et al., 2003). The main difference is that the features assigned to the second "refined topics" level are constrained by the higher

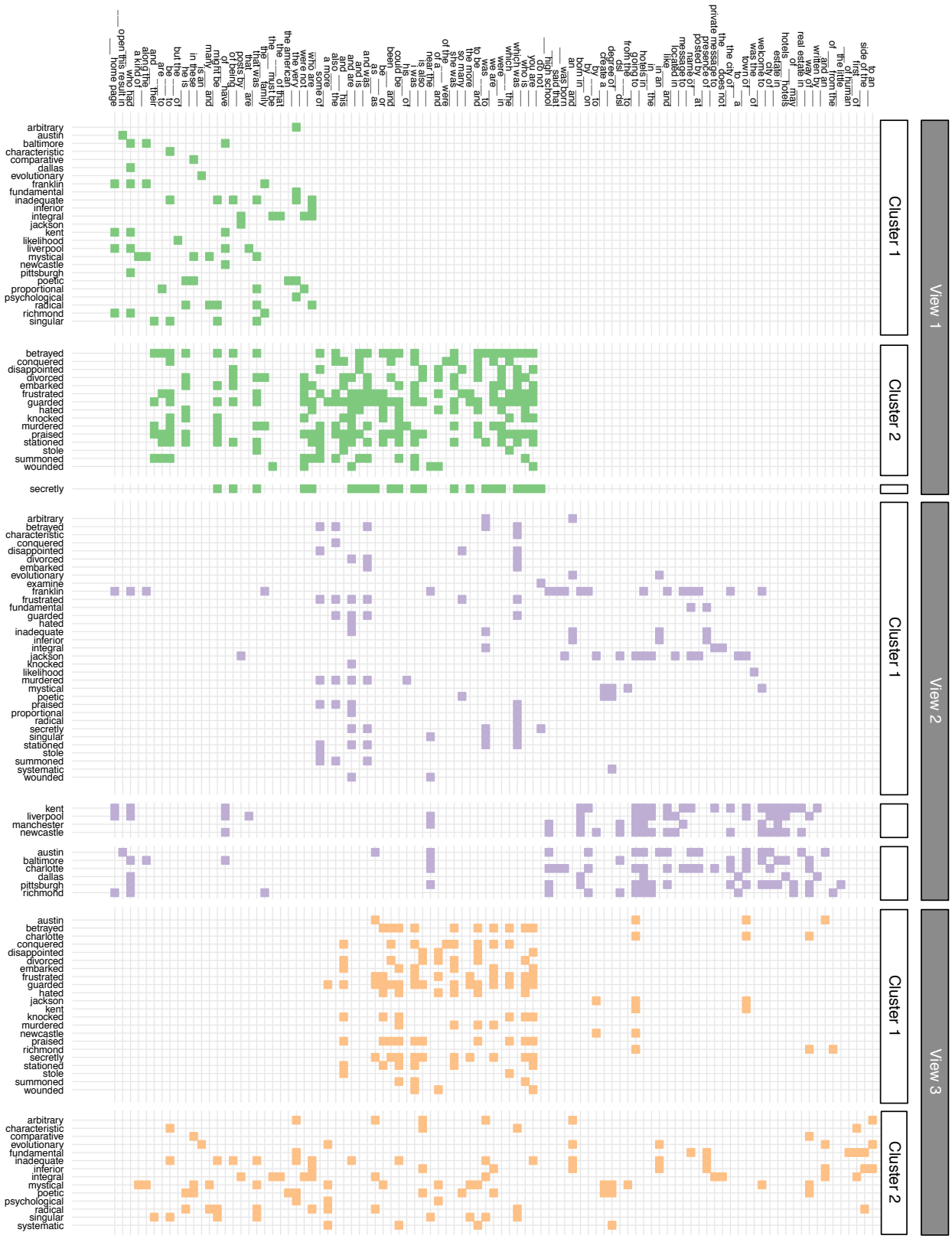


Figure 2: **Topics with Senses**: Shows top 20% of features for each view in a 3-view MVM fit to Google n-gram context data; different views place different mass on different sets of features. Cluster groupings within each view are shown. View 1 cluster 2 and View 3 cluster 1 both contain past-tense verbs, but only overlap on a subset of syntactic features.

level, similar to hierarchical clustering. Unlike hierarchical clustering, however, the top level topics/views form an admixture, allowing individual features from a single data point to be assigned to multiple views.

The most similar model to ours is *Cross-cutting categorization* (CCC), which fits multiple DPMMs to non-overlapping partitions of features (Mansinghka et al., 2009; Shafto et al., 2006). Unlike MVM, CCC *partitions* features among multiple DPMMs, hence all occurrences of a particular feature will end up in a single clustering, instead of assigning them softly using LDA. Such hard feature partitioning does not admit an efficient sampling procedure, and hence Shafto et al. (2006) rely on Metropolis-Hastings steps to perform feature assignment, making the model less scalable.

3.1 Word Representation

MVM is trained as a lexical semantic model on Web-scale n-gram and semantic context data. N-gram contexts are drawn from a combination of the Google n-gram and Google books n-gram corpora, with the head word removed: e.g. for the term *architect*, we collect contexts such as *the __ of the house*, *an __ is a*, and *the __ of the universe*. Semantic contexts are derived from word occurrence in Wikipedia documents: each document a word appears in is added as a potential feature for that word. This co-occurrence matrix is the transpose of the standard bag-of-words document representation.

In this paper we focus on two representations:

1. **Syntax-only** – Words are represented as bags of ngram contexts derived slot-filling procedure described above.

2. **Syntax+Documents** – The syntax-only representation is augmented with additional document contexts drawn from Wikipedia.

Models trained on the **syntax-only** set are only capable of capturing *syntagmatic* similarity relations, that is, words that tend to appear in similar contexts. In contrast, the **syntax+documents** set broadens the scope of modelable similarity relations, allowing for *paradigmatic* similarity (e.g. words that are topically related, but do not necessarily share common syntactic contexts).

Given such word representation data, MVM generates a fixed set of M context views corresponding to dominant eigenvectors in local syntactic or semantic space. Within each view, MVM partitions words into clusters based on each word's *local representation* in that view; that is, based on the set of context features it allocates to the view. Words have a non-uniform affinity for each view, and hence may not be present in every clustering (Figure 2). This is important as different ways of drawing distinctions between words do not necessarily apply to all words. In contrast, LDA finds locally consistent collections of contexts but does not further subdivide words into clusters given that set of contexts. Hence, it may miss more fine-grained structure, even with increased model complexity.

4 Experimental Setup

4.1 Corpora

We derive word features from three corpora: (1) the English **Google Web n-gram** corpus, containing n-gram contexts up to 5-gram that occur more than 40 times in a 1T word corpus of Web text, (2) the English **Google Books n-gram** corpus[2], consisting of n-gram contexts up to 5-gram that occur more than 40 times in a 500B word corpus of books, and (3) a snapshot of the English Wikipedia[3] taken on October 11, 2010 containing over 3M articles.

MVM is trained on a sample of 20k English words drawn uniformly at random from the top 200k English terms appearing in Wikipedia (different parts of speech were sampled from the Google n-gram corpus according to their observed frequency). Two versions of the **syntax-only** dataset are created from different subsets of the Google n-gram corpora: (1) the *common* subset contains all syntactic contexts appearing more than 200 times in the combined corpus, and (2) the *rare* subset, containing only contexts that appear 50 times or fewer.

4.2 Human Evaluation

Our main goal in this work is to find models that capture aspects of the syntactic and semantic organization of word in text that are intuitive to humans.

[2] http://ngrams.googlelabs.com/datasets
[3] http://wikipedia.org

Context Intrusion			Word Intrusion		
__ is characterized	top of the __	*country to* __	metal	dues	humor
symptoms of __	*of __ understood*	__ or less	floral	premiums	ingenuity
cases of __	along the __	__ a year	nylon	pensions	*advertisers*
in cases of __	portion of the __	__ per day	*what*	*did*	delight
real estate in __	side of the __	__ or more	ruby	damages	astonishment

Document Intrusion		
Puerto Rican cuisine	Adolf Hitler	History of the Han Dynasty
Greek cuisine	*List of General Hospital characters*	Romance of the Three Kingdoms
ThinkPad	History of France	*List of dog diseases*
Palestinian cuisine	Joachim von Ribbentrop	Conquest of Wu by Jin
Field ration	World War I	Mongolia

Table 1: Example questions from the three intrusion tasks, in order of difficulty (left to right, easy to hard; computed from inter-annotator agreement). *Italics* show intruder items.

According to the *use theory* of meaning, lexical semantic knowledge is equivalent to knowing the contexts that words appear in, and hence being able to form reasonable hypotheses about the relatedness of syntactic contexts.

Vector space models are commonly evaluated by comparing their similarity predictions to a nominal set of human similarity judgments (Curran, 2004; Padó and Lapata, 2007; Schütze, 1998; Turney, 2006). In this work, since we are evaluating models that potentially yield many different similarity scores, we take a different approach, scoring clusters on their semantic and syntactic *coherence* using a *set intrusion* task (Chang et al., 2009).

In set intrusion, human raters are shown a set of options from a coherent group and asked to identify a single *intruder* drawn from a different group. We extend intrusion to three different lexical semantic tasks: (1) *context intrusion*, where the top contexts from each cluster are used, (3) *document intrusion*, where the top document contexts from each cluster are used, and (2) *word intrusion*, where the top words from each cluster are used. For each cluster, the top four contexts/words are selected and appended with another context/word from a different cluster.[4] The resulting set is then shuffled, and the human raters are asked to identify the intruder, after being given a short introduction (with common examples) to the task. Table 1 shows sample questions of varying degrees of difficulty. As the semantic coherence and distinctness from other clusters increases, this task becomes easier.

Set intrusion is a more robust way to account for human similarity judgments than asking directly for a numeric score (e.g., the Miller and Charles (1991) set) as less calibration is required across raters. Furthermore, the additional cluster context significantly reduces the variability of responses.

Human raters were recruited from *Amazon's Mechanical Turk*. A total of 1256 raters completed 30438 evaluations for 5780 unique intrusion tasks (5 evaluations per task). 2736 potentially fraudulent evaluations from 11 raters were rejected.[5] Table 3 summarizes inter-annotator agreement. Overall we found $\kappa \approx 0.4$ for most tasks; a set of comments about the task difficulty is given in Table 2, drawn from an anonymous public message board.

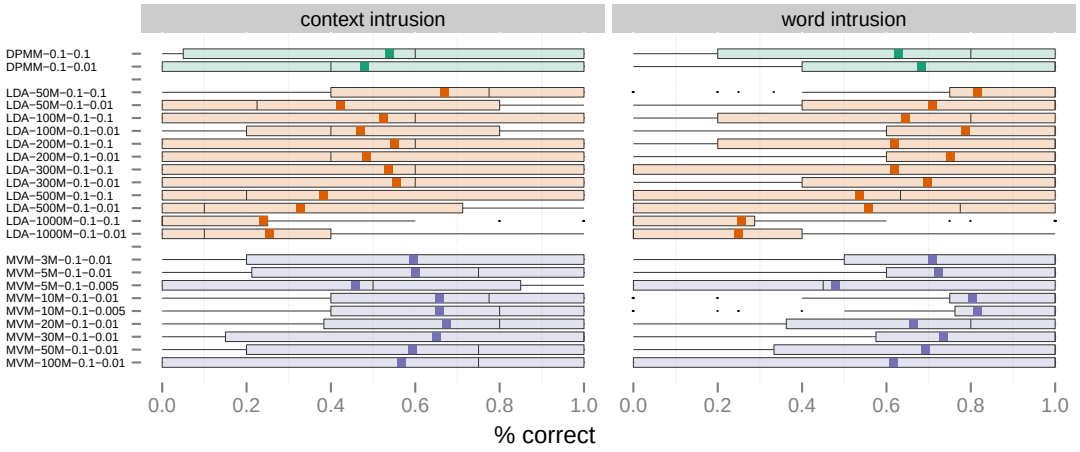
5 Results

We trained DPMM, LDA and MVM models on the **syntax-only** and **syntax+documents** data across a wide range of settings for $M \in \{3, 5, 7, 10, 20, 30, 50, 100, 200, 300, 500, 1000\}$,[6]

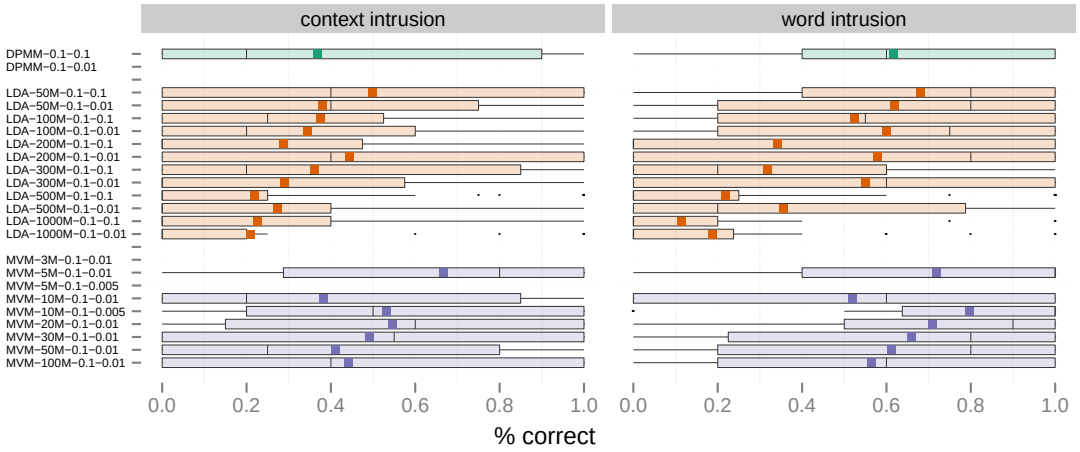
[4] Choosing four elements from the cluster uniformly at random instead of the top by probability led to lower performance across all models.

[5] (**Rater Quality**) Fraudulent Turkers were identified using a combination of average answer time, answer entropy, average agreement with other raters, and adjusted answer accuracy.

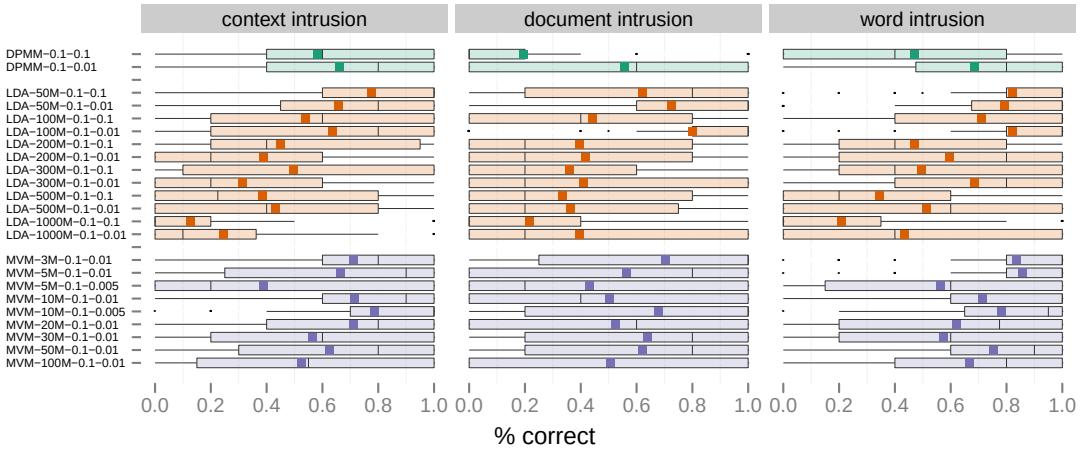
[6] LDA is run on a different range of M settings from MVM (50-1000 vs 3-100) in order to keep the effective number of



(a) **Syntax-only**, common n-gram contexts.



(b) **Syntax-only**, rare n-gram contexts.



(c) **Syntax+Documents**, common n-gram contexts.

Figure 3: Average scores for each model broken down by parameterization and data source. Error bars depict 95% confidence intervals. X-axis labels show **Model-views**-α-β. Dots show average rater scores; bar-charts show standard quantile ranges and median score.

U1	I just tried 30 of the what doesn't belong ones. They took about 30 seconds each due to thinking time so not worth it for me.
U2	I don't understand the fill in the blank ones to be honest. I just kinda pick one,since I don't know what's expected lol
U3	Your not filling in the blank just ignore the blank and think about how the words they show relate to each other and choose the one that relates least. Some have just words and no blanks.
U4	These seem very subjective to mw. i hope there isn't definite correct answers because some of them make me go [emoticon of head-scratching]
U5	I looked and have no idea. I guess I'm a word idiot because I don't see the relation between the words in the preview HIT - too scared to try any of these.
U6	I didn't dive in but I did more than I should have they were just too easy. Most of them I could tell what did not belong, some were pretty iffy though.

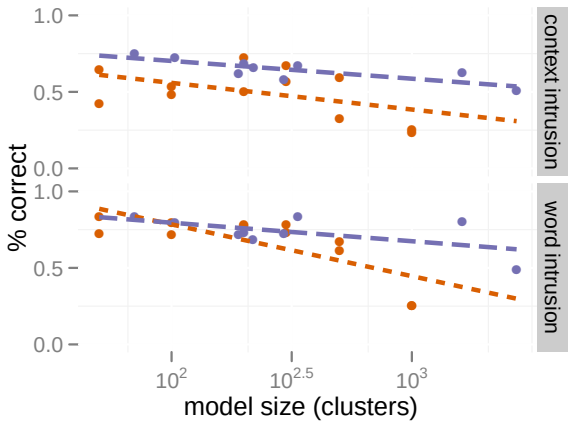
Table 2: Sample of comments about the task taken verbatim from a public Mechanical Turk user message board (TurkerNation). Overall the raters report the task to be difficult, but engaging.

$\alpha \in \{0.1, 0.01\}$, and $\beta \in \{0.1, 0.05, 0.01\}$ in order to understand how they perform relatively on the intrusion tasks and also how sensitive they are to various parameter settings.[7] Models were run until convergence, defined as no increase in log-likelihood on the training set for 100 Gibbs samples. Average runtimes varied from a few hours to a few days, depending on the number of clusters or topics. There is little computational overhead for MVM compared to LDA or DPMM with a similar number of clusters.

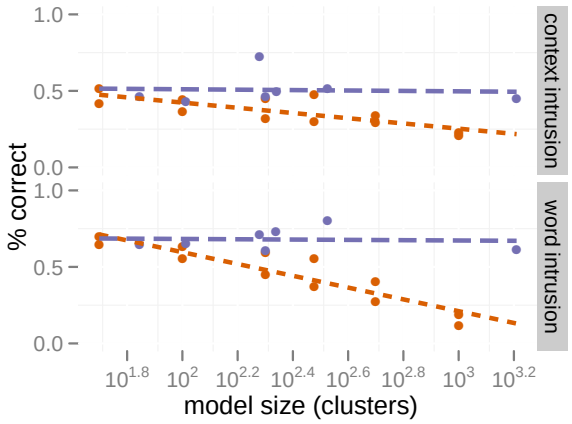
Overall, MVM significantly outperforms both LDA and DPMM (measured as % of intruders correctly identified) as the number of clusters increases. Coarse-grained lexical semantic distinctions are easy for humans to make, and hence models with fewer clusters tend to outperform models with more clusters. Since high granularity predictions are more useful for downstream tasks, we focus on the interplay between model complexity and performance.

clusters (and hence model capacity) roughly comparable.

[7]We did not compare directly to Cross-cutting categorization, as the Metropolis-Hasting steps required that model were too prohibitively expensive to scale to the Google n-gram data.



(a) **Syntax-only**, common n-gram contexts.



(b) **Syntax-only**, rare n-gram contexts.

Figure 4: Scatterplot of model size vs. avg score for MVM (dashed, purple) and LDA (dotted, orange).

5.1 Syntax-only Model

For common n-gram context features, MVM performance is significantly less variable than LDA on both the word intrusion and context intrusion tasks, and furthermore significantly outperforms DPMM (Figure 3(a)). For context intrusion, DPMM, LDA, and MVM average 57.4%, 49.5% and 64.5% accuracy respectively; for word intrusion, DPMM, LDA, and MVM average 66.7%, 66.1% and 73.6% accuracy respectively (averaged over all parameter settings). These models vary significantly in the average number of clusters used: 373.5 for DPMM, 358.3 for LDA and 639.8 for MVM, i.e. the MVM model is signifi-

Model	Syntax	Syntax+Documents	Overall
DPMM	0.30	0.40	**0.33**
LDA	0.33	0.39	**0.35**
MVM	0.44	0.49	**0.46**
Overall	**0.37**	**0.43**	**0.39**

Table 3: Fleiss' κ scores for various model and data combinations. Results from MVM have higher κ scores than LDA or DPMM; likewise **Syntax+Documents** data yields higher agreement, primarily due to the relative ease of the document intrusion task.

cantly more granular. Figure 4(a) breaks out model performance by model complexity, demonstrating that MVM has a significant edge over LDA as model complexity increases.

For rare n-gram contexts, we obtain similar results, with MVM scores being less variable across model parameterizations and complexity (Figure 3(b)). In general, LDA performance degrades faster as model complexity increases for rare contexts, due to the increased data sparsity (Figure 4(b)). For context intrusion, DPMM, LDA, and MVM average 45.9%, 36.1% and 50.9% accuracy respectively; for word intrusion, DPMM, LDA, and MVM average 67.4%, 45.6% and 67.9% accuracy; MVM performance does not differ significantly from DPMM, but both outperform LDA. Average cluster sizes are more uniform across model types for rare contexts: 384.0 for DPMM, 358.3 for LDA and 391 for MVM.

Human performance on the context intrusion task is significantly more variable than on the word-intrusion task, reflecting the additional complexity.

In all models, there is a high correlation between rater scores and per-cluster likelihood, indicating that model confidence reflects noise in the data.

5.2 Syntax+Documents Model

With the **syntax+documents** training set, MVM significantly outperforms LDA across a wide range of model settings. MVM also outperforms DPMM for word and document intrusion. For context intrusion, DPMM, LDA, and MVM average 68.0%, 51.3% and 66.9% respectively;[8] for word intrusion, DPMM, LDA, and MVM average 56.3%, 64.0% and 74.9% respectively; for document intrusion, DPMM, LDA,

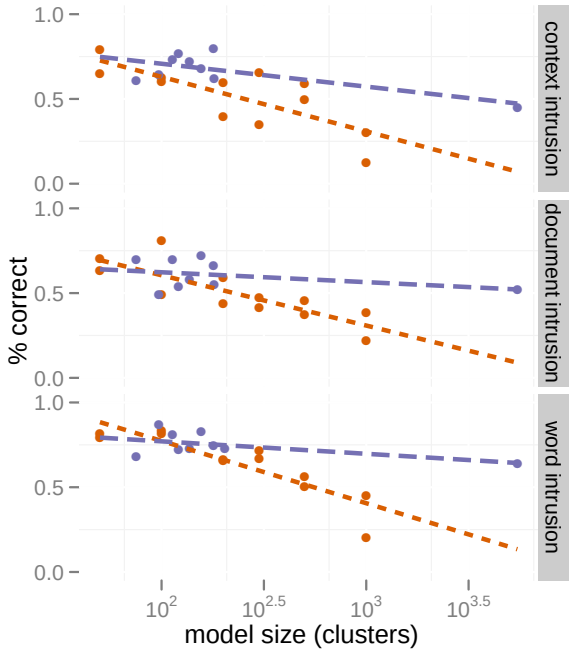


Figure 5: Scatterplot of model size vs. avg score for MVM (dashed, purple) and LDA (dotted, orange); **Syntax+Documents** data.

and MVM average 41.5%, 49.7% and 60.6% respectively. Qualitatively, models trained on **syntax+document** yield a higher degree of paradigmatic clusters which have intuitive thematic structure. Performance on document intrusion is significantly lower and more variable, reflecting the higher degree of world knowledge required. As with the previous data set, performance of MVM models trained on **syntax+documents** data degrades less slowly as the cluster granularity increases (Figure 5).

One interesting question is to what degree MVM *views* partition syntax and document features versus LDA topics. That is, to what degree do the MVM views capture purely syntagmatic or purely paradigmatic variation? We measured *view entropy* for all three models, treating syntactic features and document features as different class labels. MVM with $M = 50$ views obtained an entropy score of 0.045, while LDA with $M = 50$ obtained 0.073, and the best DPMM model 0.082.[9] Thus MVM views may indeed capture pure syntactic or thematic clusterings.

[8] High DPMM accuracy is driven by the low number of clusters: 46.5 for DPMM vs. 358.3 for LDA and 725.6 for MVM.

[9] The low entropy scores reflect the higher percentage of syntactic contexts overall.

5.3 Discussion

As cluster granularity increases, we find that MVM accounts for feature noise better than either LDA or DPMM, yielding more coherent clusters. (Chang et al., 2009) note that LDA performance degrades significantly on a related task as the number of topics increases, reflecting the increasing difficulty for humans in grasping the connection between terms in the same topic. This suggests that as topics become more ne-grained in models with larger number of topics, they are less useful for humans. In this work, we find that although MVM and LDA perform similarity on average, MVM clusters are significantly more interpretable than LDA clusters as the granularity increases (Figures 4 and 5). We argue that models capable of making such fine-grained semantic distinctions are more desirable.

The results presented in the previous two sections hold both for unbiased cluster selection (e.g. where clusters are drawn uniformly at random from the model) *and* when cluster selection is biased based on model probability (results shown). Biased selection potentially gives an advantage to MVM, which generates many more small clusters than either LDA or DPMM, helping it account for noise.

6 Future Work

Models based on cross-cutting categorization is a novel approach to lexical semantics and hence should be evaluated on standard baseline tasks, e.g. contextual paraphrase or *lexical substitution* (McCarthy and Navigli, 2007). Additional areas for future work include:

(**Latent Relation Modeling**) Clusterings formed from feature partitions in MVM can be viewed as a form of *implicit* relation extraction; that is, instead of relying on explicit surface patterns in text, relations between words or concepts are identified indirectly based on common syntactic patterns. For example, clusterings that divide cities by geography or clusterings partition adjectives by their polarity.

(**Latent Semantic Language Modeling**) Generative models such as MVM can be used to build better priors for class-based language modeling (Brown et al., 1992). The rare n-gram results demonstrate that MVM is potentially useful for tail contexts; i.e. inferring tail probabilities from low counts.

(**Explicit Feature Selection**) In this work we rely on smoothing to reduce the noise of over-broad extraction rather than performing feature selection explicitly. All of the models in this paper can be combined with feature selection methods to remove noisy features, and it would be particularly interesting to draw parallels to models of "clutter" in vision.

(**Hierarchical Cross-Categorization**) Human concept organization consists of multiple overlapping local ontologies, similar to the loose ontological structure of Wikipedia. Furthermore, each ontological system has a different set of salient properties. It would be interesting to extend MVM to model hierarchy explicitly, and compare against baselines such as *Brown clustering* (Brown et al., 1992), the nested Chinese Restaurant Process (Blei et al., 2003) and the hierarchical Pachinko Allocation Model (Mimno et al., 2007).

7 Conclusion

This paper introduced MVM, a novel approach to modeling lexical semantic organization using multiple *cross-cutting* clusterings capable of capturing multiple lexical similarity relations jointly in the same model. In addition to robustly handling homonymy and polysemy, MVM naturally captures both *syntagmatic* and *paradigmatic* notions of word similarity. MVM performs favorably compared to other generative lexical semantic models on a set of human evaluations, over a wide range of model settings and textual data sources.

Acknowledgements

We would like to thank the anonymous reviewers for their extensive comments. This work was supported by a Google PhD Fellowship to the first author.

References

David Blei, Thomas Griffiths, Michael Jordan, and Joshua Tenenbaum. 2003. Hierarchical topic models and the nested Chinese restaurant process. In *Proc. NIPS-2003*.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *NIPS*.

James Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proc. of the ACL*. Association for Computer Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proc. of WWW 2001*.

James Gorman and James R. Curran. 2006. Scaling distributional similarity to large corpora. In *Proc. of ACL 2006*.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114:2007.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Vikash K. Mansinghka, Eric Jonas, Cap Petschulat, Beau Cronin, Patrick Shafto, and Joshua B. Tenenbaum. 2009. Cross-categorization: A method for discovering multiple overlapping clusterings. In *Proc. of Nonparametric Bayes Workshop at NIPS 2009*.

Diana McCarthy and Roberto Navigli. 2007. SemEval-2007 task 10: English lexical substitution task. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics.

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

David Mimno, Wei Li, and Andrew McCallum. 2007. Mixtures of hierarchical topics with pachinko allocation. In *ICML*.

Gregory L. Murphy. 2002. *The Big Book of Concepts*. The MIT Press.

Donglin Niu, Jennifer G. Dy, and Michael I. Jordan. 2010. Multiple non-redundant spectral clustering views. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 831–838.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Joseph Reisinger and Raymond J. Mooney. 2010. A mixture model with sharing for lexical semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics*, pages 52–57. ACL.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Patrick Shafto, Charles Kemp, Vikash Mansinghka, Matthew Gordon, and Joshua B. Tenenbaum. 2006. Learning cross-cutting systems of categories. In *Proc. CogSci 2006*.

Rion Snow, Daniel Jurafsky, and Andrew Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proc. of ACL 2006*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of the ACL*.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Amos Tversky and Itamar Gati. 1982. Similarity, separability, and the triangle inequality. *Psychological Review*, 89(2):123–154.

Benjamin Van Durme and Marius Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proc. of AAAI 2008*.

Reducing Grounded Learning Tasks to Grammatical Inference

Benjamin Börschinger
Department of Computing
Macquarie University
Sydney, Australia
benjamin.borschinger@mq.edu.au

Bevan K. Jones
School of Informatics
University of Edinburgh
Edinburgh, UK
b.k.jones@sms.ed.ac.uk

Mark Johnson
Department of Computing
Macquarie University
Sydney, Australia
mark.johnson@mq.edu.au

Abstract

It is often assumed that 'grounded' learning tasks are beyond the scope of grammatical inference techniques. In this paper, we show that the grounded task of learning a semantic parser from ambiguous training data as discussed in Kim and Mooney (2010) can be reduced to a Probabilistic Context-Free Grammar learning task in a way that gives state of the art results. We further show that additionally letting our model learn the language's canonical word order improves its performance and leads to the highest semantic parsing f-scores previously reported in the literature.[1]

1 Introduction

One of the most fundamental ideas about language is that we use it to express our thoughts. Learning a natural language, then, amounts to (at least) learning a mapping between the things we utter and the things we think, and can therefore be seen as the task of learning a semantic parser, i.e. something that maps natural language expressions such as sentences into meaning representations such as logical forms. Obviously, this learning can neither take place in a fully supervised nor in a fully unsupervised fashion: the learner does not 'hear' the meanings of the sentences she observes, but she is also not treating them as merely meaningless strings. Rather, it seems plausible to assume that she uses extra-linguistic context to assign certain meanings to the linguistic input she is confronted with.

In this sense, learning a semantic parser seems to go beyond the well-studied task of unsupervised grammar induction. It involves not only learning a grammar for the form-side of language, i.e. language expressions such as sentences, but also the 'grounding' of this structure in meaning representations. It requires going beyond the mere linguistic input to incorporate, for example, perceptual information that provides a clue to the meaning of the observed forms. Essentially, it seems as if 'grounded' learning tasks like this require dealing with two different kinds of information, the purely formal (phonemic) and meaningful (semantic) aspects of language. Grammatical inference seems to be limited to dealing with one level of formal information (Chang and Maia, 2001). For this reason, probably, approaches to the task of learning a semantic parser employ a variety of sophisticated and task-specific techniques that go beyond (but often elaborate on) the techniques used for grammatical inference (Lu et al., 2008; Chen and Mooney, 2008; Liang et al., 2009; Kim and Mooney, 2010; Chen et al., 2010).

In this paper, we show that one can reduce the task of learning a semantic parser to a Probabilistic Context Free Grammar (PCFG) learning task, and more generally, that grounded learning tasks are not in principle beyond the scope of grammatical inference techniques. In particular, we show how to formulate the task of learning a semantic parser as discussed by Chen, Kim and Mooney (2008, 2010) as the task of learning a PCFG from strings. Our model does not only constitute a proof of concept that this

[1] The source code used for our experiments and the evaluation is available as supplementary material for this article.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1416–1425,
Edinburgh, Scotland, UK, July 27–31, 2011.

reduction is possible for certain cases, it also yields highly competitive results.[2]

By reducing the problem to the well understood PCFG formalism, it also becomes easy to consider extensions, leading to our second contribution. We demonstrate that a slight modification to our model so that it also learns the language's canonical word order improves its performance even beyond the best results previously reported in the literature. This language-independent and linguistically well motivated elaboration allows the model to learn a global fact about the language's syntax, its canonical word order.

Our contribution is two-fold. We provide an illustration of how to reduce grounded learning tasks to grammatical inference. Secondly, we show that extending the model so that it can learn linguistically well motivated generalizations such as the canonical word order can lead to better results.

The structure of the paper is as follows. First we give a short overview of the previous work by Chen, Kim and Mooney and describe their dataset. Then, we show how to reduce the parsing task addressed by them to a PCFG-learning task. Finally, we explain how to let our model additionally learn the language's canonical word order.

2 Previous Work by Chen, Kim and Mooney

In a series of recent papers, Chen, Kim and Mooney approach the task of learning a semantic parser from *ambiguous* training data (Chen and Mooney, 2008; Kim and Mooney, 2010; Chen et al., 2010). This goes beyond previous work on semantic parsing such as Lu et al. (2008) or Zettlemoyer and Collins (2005) which rely on *unambiguous* training data where every sentence is paired only with its meaning. In contrast, Chen, Kim and Mooney allow their training examples to exhibit the kind of uncertainty about sentence meanings human learners are likely to have to deal with by allowing for sentences to be associated with a *set* of candidate-meanings, and the correct meaning might not even be in this set. They create the training data by first collecting humanly generated written language comments on four different RoboCup games. The comments are recorded with a time-stamp and then associated with all game events automatically extracted from the games which occured up to five seconds before the comment was made. This leads to an ambiguous pairing of comments with candidate meanings that can be considered similar to the "linguistic input in the context of a rich, relevant, perceptual environment" to which real language learners probably have access (Chen and Mooney, 2008). For evaluation purposes, they manually create a gold-standard which contains unambiguous natural language comment / event pairs. Due to the fact that some comments refer to events not detected by their extraction-algorithm, not every natural language sentence has a gold matching meaning representation. In addition to the inherent ambiguity of the training examples, the learner therefore has to somehow deal with those examples which only have 'wrong' meanings associated with them.

Datasets exist for both Korean and English, each comprising training and gold data for four games.[3] Some details about this data are given in Table 1, such as the number of examples, their average ambiguity and the number of misleading examples.

For the following short discussion of previous approaches, we mainly focus on Kim and Mooney (2010). This is the most recent publication and reports the highest scores.

2.1 The parsing task

Learning a semantic parser from the ambiguous data is, in fact, just one of three tasks discussed by Kim and Mooney (2010), henceforth KM. In addition to parsing, they discuss matching and natural language generation. We are ignoring the generation task as we are currently only interested in the parsing problem, and we treat the matching task, picking the correct meaning from the set of candidates, merely as a byproduct of parsing, rather than as a completely separate task: parsing implicitly requires the model to disambiguate the data it is learning from.

[2] It has been pointed out to us by one reviewer that the task we address falls short of what is often called 'grounded learning'. We acknowledge that semantic parsing constitutes a very limited kind of grounded learning but want to point out that the task has been introduced as an instance of grounded learning in the previous literature such as Chen and Mooney (2008).

[3] The datasets are freely available at `http://www.cs.utexas.edu/~ml/clamp/sportscasting/`. We retrieved the data used here on March 29th, 2011.

	Number of comments				Ambiguity	
	# Training	# Training with Gold Match	# Training with correct MR	# Gold	Noise	Avg. # of MRs
English dataset						
total	1872	1492	1360	1539	0.2735	2.20
Korean dataset						
total	1914	1763	1733	1763	0.0946	2.39

Table 1: Statistics for the Korean and the English datasets. The numbers are basically identical to those reported in Chen et al. (2010) except for minimal differences in the number of training examples (we give one more for every English training set, and one more for the 2004 Korean training set). In addition, our calculation of the average sentential ambiguity (Avg. # of MRs) differs because we assume that mutiple occurences of the same event in a context do not add to the overall ambiguity, and our calculation of the noise (fraction of training examples without the correct meaning in their context) takes into account that there are training examples which do not have their gold meaning associated with them in the training data and is therefore slightly higher than the one reported in Chen et al. (2010).

KM's model builds on previous work by Lu et al. (2008) and is a generative model which defines a joint probability distribution over natural language sentences (NLs), meaning representations (MRs) and hybrid trees. The NLs are the natural language comments to the games, the MRs are simple logical formulae describing game events and playing the role of sentence meanings, and a hybrid tree is a tree structure that represents the correspondence between a sentence and its meaning. More specifically, if some NL w has as its meaning an MR m, and m has been generated by a meaning grammar (MG) G, the hybrid tree corresponding to the pair $\langle$w,m$\rangle$ has as its internal nodes those rules of G used in the derivation of m, and as its leaves the words making up w.[4] An example hybrid tree for the pair $\langle$THE PINK GOALIE PASSES THE BALL TO PINK11,pass(pink1,pink11)$\rangle$ is given in Figure 1. Their model is trained by a variant of the Inside-Outside algorithm which deals with the hybrid tree structure and takes into account the ambiguity of the training examples.

In addition to learning directly from the ambiguous training data, they also train a semantic parser in a supervised fashion on data that has been previously disambiguated by their matching model. This slightly improves their system's performance. Consequently, there are two scores for each of the two languages (English and Korean) with which we compare our own model: those of the parsers trained directly from the ambiguous data, and those of the 'supervised' parsers which constitute the current state of the art. The details of their evaluation method are disccused in Section 3.3, and their scores are given in Table 2, together with our own scores.

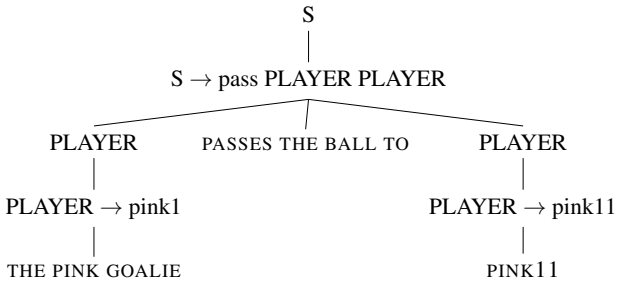


Figure 1: A hybrid tree for the sentence-meaning pair $\langle$THE PINK GOALIE PASSES THE BALL TO PINK11,pass(pink1,pink11)$\rangle$. The internal nodes correspond to the rules used to derive pass(pink1,pink11) from a given Meaning Grammar, and the leaves correspond to the words or substrings that make up the sentence.

3 Learning a Semantic Parser as a PCFG-learning problem

Given that one can effectively represent both a sentence's form and its meaning in a hybrid tree, it is interesting to ask whether one can do with a structure that can be learned by grammatical inference tech-

[4] We use SMALL CAPS for words, sans serif for MRs and MR constituents (concepts), and *italics* for non-terminals and Grammars.

niques from strings which incorporate the contextual information. In this section, we show how to reduce hybrid trees to such 'standard' trees. In effect, we show via construction that 'grounded' learning tasks such as learning a semantic parser from semantically enriched and ambiguous data can be reduced to 'ungrounded' tasks such as grammatical inference.

Instead of taking the internal nodes of the trees generated by our model as corresponding to MG *production rules*, we take them to correspond to MR *constituents*. The MR pass(pink1,pink11), for example, has 4 constituents: the whole MR, the predicate pass, and the two arguments pink1 and pink11. Figure 2 gives the tree we assume instead of Figure 1 for the sentence-meaning pair ⟨THE PINK GOALIE PASSES THE BALL TO PINK11,pass(pink1,pink11)⟩. Its root is assumed to correspond to the whole MR and is labeled $S_{pass(pink1,pink11)}$. The remaining three MR constituents correspond to the root's daughters which we label $Phrase_{pink1}$, $Phrase_{pass}$ and $Phrase_{pink11}$. Generally speaking, we assume a special non-terminal S_m for every MR m generated by the MG, and a special non-terminal $Phrase_{con}$ for each of the terminals of the MG (which loosely correspond to concepts). This is only possible for MGs which create a finite set of MRs, but the MG used by Kim and Mooney (2010) obeys this restriction.[5]

The tree's terminals are the words that make up the sentence, and we assume them to be dominated by concept-specific pre-terminals $Word_{con}$ which correspond to concept-specific probability distributions over the language's vocabulary. Since each $Phrase_{con}$ may span multiple words, we give trees rooted in $Phrase_{con}$ a left-recursive structure that corresponds to a unigram Markov-process. This process generates an arbitrary sequence of words semantically related to con, dominated by the corresponding pre-terminal $Word_{con}$ in our model, and words not directly semantically related to con, dominated by a special word pre-terminal $Word_{\emptyset}$. The sole further restriction is that every $Phrase_{con}$ must contain at least one $Word_{con}$.

Trees like the one in Figure 2 can be generated by a Context-Free Grammar (CFG) which, in turn, can be trained on strings to yield a PCFG which embodies a semantic parser as will be discussed in Section 3.3. We now describe how to set up such a CFG in a systematic way and how to train it on the data used by KM.

3.1 Setting up the PCFG

The training data expresses information of two different kinds – form and meaning. Every training example consists of a natural language string (the formal information) and a set of candidate meanings for the string (the semantic information, its context), allowing for the possibility that none of the meanings in the context is the correct one. In order to learn from data like this within a grammatical inference framework, we have to encode the semantic information as part of the string. Assigning a specific MR m to a string corresponds, in our framework, to analyzing it as a tree with S_m as its root. A sentence's context constrains which of the many possible meanings might be expressed by the string. Thus the role played by the context is adequately modelled if we ensure that if a string w is associated with a context $\{m_1,...,m_n\}$, the model only considers the possibilities that this string might be analyzed as $S_{m_1},...,S_{m_n}$.

There are 959 different contexts, i.e. 959 different sets of MRs, in the English data set (984 for the Korean data), and we therefore introduce 959 new terminal symbols which play the role of context-identifiers, for example c_1 to c_{959}.[6] Formally speaking, a context-identifier is a terminal like any other word of the language and we can therefore prefix every comment in the training data with the context-identifier standing for the set of MRs associated with this comment, an idea taken from previous work such as Johnson et al. (2010). Thus having incorporated the contextual information into the string, we go on to show how our model makes use of this information, considering the MR pass(pink1,pink11) as an example. A formal description of the model is given Figure 3.

Assume that pass(pink1,pink11) is associated with only one training example and therefore occurs only in one specific context. If the context-identifier introduced for this context is c_1, we require the

[5]This grammar is given in the Appendix to Chen et al. (2010) and generates a total of 2048 MRs.

[6]If we were to consider every possible context, we would have to consider 2^{2048} contexts because the MG generates 2048 MRs.

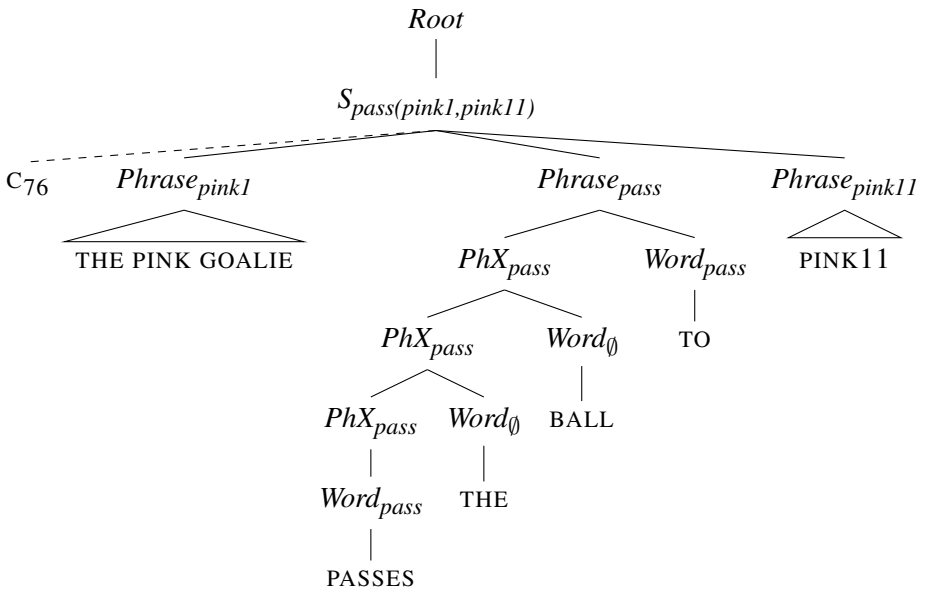


Figure 2: The tree-structure we propose instead of the Hybrid Tree structure used by (Kim and Mooney, 2010). The non-terminal nodes do not correspond to MG productions, but to MR constituents. The internal structure of the $Phrase_{con}$ constituents, shown in full detail for $Phrase_{pass}$, corresponds to a Markov process that generates the words that make up the sentence. The terminal C_{76} is a context-identifier that restricts the range of S_m non-terminals that might dominate the sentence and is only used during training, as described in Section 3.1. The grammar that generates this trees is described in Figure 3.

right-hand side of all rules with $S_{pass(pink1,pink11)}$ on their left-hand side to begin with C_1. More generally, if an MR m occurs in the contexts associated with the context-identifiers C_K,...,C_L, we require the right-hand side of all rules with S_m on their left-hand side to begin with exactly one of C_K,...,C_L.

In this sense, the *context-identifiers* can be seen as providing the model with a *top-down constraint* – if it encounters a context-identifier, it can only try analyses leading to MRs which are licensed by this context-identifier. On the other hand, the words have to be generated by concept-specific word-distributions, and the concepts that are present restrict the range of possible S_m non-terminals which might dominate the whole string. In this sense, the *words* the model observes provide it with a *bottom-up constraint* – if it sees words which are semantically related to certain concepts con_1,...,con_n, it has to arrive at an MR which licenses the presence of the corresponding $Phrase_{con_x}$ non-terminals. Of course, the model has to also learn which words are semantically related to which concepts. To enable it to do this, our grammar allows every $Word_x$ non-terminal to be rewritten as every word of the language.

Since there are sentences in the training data without the correct meaning in their context, we want to give our model the possibility of not assigning to a sentence any of the MRs licensed by its context-identifier. To do this, we employ another trick of previous work by Johnson et. al and assume a special null meaning $\emptyset$ to be present in every context. $S_\emptyset$ may only span words generated by $Word_\emptyset$, the language-specific distribution for words not directly related to any concept; this also has to be learned by the model.

As a last complication, we deal with the fact that syntactic constituents are linearized with respect to each other. For example, if an MR has 3 proper constituents (i.e. excluding the MR itself), our grammar allows the corresponding 3 syntactic constituents – which we might label $Phrase_{predicate}$, $Phrase_{arg1}$ and $Phrase_{arg2}$ – to occur in any of the 6 possible orders. Therefore, we have an S_m rule for every context in which m occurs and for every possible order of the proper constituents of m.

A formally explicit description of the rule

schemata used to generate the CFG is given in Figure 3.[7] Instantiating all those schemata leads to a grammar with 33,101 rules for the English data and 30,731 rules for the Korean data. The difference in size is due to differences in the size of the vocabulary and the different number of contexts in the data sets.

These CFGs can now be trained on the training data using the Inside-Outside algorithm (Lari and Young, 1990). After training, the resulting PCFG embodies a semantic parser in the sense that, with a slight modification we describe in section 3.3, it can be used to parse a string into its meaning representation by determining the most likely syntactic analysis and reading off the meaning assigned by our model at the S_m-node.

3.2 Possible objections to our reduction

Before we go on to discuss the details of training and evaluation of our model, we want to address an objection that might seem tempting. Isn't our reduction impractical and unrealistic as even a highly abstract model of language learning – after all, setting up the huge CFG requires knowledge about the vocabulary, the MG and all the complicated rules discussed which, presumably, is more knowledge than we want to provide a language learner with, lest we trivialize the task. To this we reply firstly, that it is true that our reduction only works for *offline* or *batch grounded learning tasks* where all the data is available to the model before the actual learning begins so that it 'knows' the words, the meanings and the contexts present in the data. This offline constraint is, however, true of all models which are trained by iterating multiple times over training data such as KM's model. Secondly, the intimidating CFG can in principle be reduced to a hand-full of intuitive principles and is easy to generate automatically.

First of all, the many specific S_m-rewrite rules reduce to the heuristic that every semantic constituent should correspond to a syntactic constituent, and the fact that natural language expressions are linearly ordered. Note that our model does not contain knowledge about the specific word order of the language. It simply allows for the constituents of an MR to occur in every possible order which is a very unbiased and empiricist assumption. Of course, this leads to some limited kind of 'implicit learning' of word order in the sense that for every meaning and for every context, our model might (and in most cases will) assign different probabilities to the different rules for every word order; so it can learn that certain specific MRs such as pass(pink1,pink11) are more often linearized in one way than in any other. It cannot, however, generalize this to other (or even unseen) MRs, i.e. it does not learn a global fact about the language. In a way, it lacks the knowledge that there is such a thing as word order, a point which we will elaborate on in Section 4.

The many re-write rules for the pre-terminal $Word_x$s are nothing but an explicit version of the assumption that every word the model encounters might, in principle, be semantically related to every concept it knows. Again, this seems to us to be a reasonable assumption.

Finally, the complicated looking set of rules for the internal structure of $Phrase_x$s corresponds to a simple unigram Markov-process for generating strings. All in all, we do not see that we make any more assumptions than other approaches; our formulation may make explicit how rich those assumptions are but we have not qualitatively changed them.

3.3 Training and Evaluation

The CFG described in the previous section is trained on the same training data used by KM, except that we reduce it to strings (without changing the information present in the original data) by prefixing every sentence with a context-identifier. For training we run the Inside-Outside algorithm[8] with uniform initialization weights until convergence. For English, this results in an average number of 76 iterations for each fold, for Korean the average number of iterations is 50. To deal with the fact that the model might not observe certain meanings during training, we apply a simple smoothing technique by using a Dirichlet prior of α=0.1 on the rule probabilities. In effect, this provides our system with a small number of pseudo-observations for each rule which prevents

[7] In our description, we use context-identifiers such as c_1 with a systematic ambiguity, letting them stand for the terminal symbol representing a context and, in contexts such as $m \in c_1$, for the represented context itself.

[8] We use Mark Johnson's freely available implementation, available at `http://web.science.mq.edu.au/~mjohnson/Software.htm`.

$\text{Root} \rightarrow S_m$	$m \in M \cup \{\emptyset\}$
$S_m \rightarrow c\, Phrase_{p(m)}$	$c \in C, m \in c, m \in Pred0(M)$
$S_m \rightarrow c\, \{Phrase_{p(m)}, Phrase_{a1(m)}\}$	$c \in C, m \in c, m \in Pred1(M)$
$S_m \rightarrow c\, \{Phrase_{p(m)}, Phrase_{a1(m)}, Phrase_{a2(m)}\}$	$c \in C, m \in c, m \in Pred2(M)$
$S_\emptyset \rightarrow c\, Phrase_\emptyset$	$c \in C$
$Phrase_\emptyset \rightarrow Word_\emptyset$	
$Phrase_\emptyset \rightarrow Phrase_\emptyset\, Word_\emptyset$	
$Phrase_x \rightarrow Word_x$	$x \in T$
$Phrase_x \rightarrow PhX_x\, Word_x$	$x \in T$
$Phrase_x \rightarrow Ph_x\, Word_\emptyset$	$x \in T$
$PhX_x \rightarrow Word_r$	$x \in T, r \in \{x, \emptyset\}$
$PhX_x \rightarrow PhX_x\, Word_r$	$x \in T, r \in \{x, \emptyset\}$
$Ph_x \rightarrow PhX_x\, Word_x$	$x \in T$
$Ph_x \rightarrow Ph_x\, Word_\emptyset$	$x \in T$
$Ph_x \rightarrow Word_x$	$x \in T$
$Word_x \rightarrow v$	$x \in T \cup \{\emptyset\}, v \in V$

Figure 3: The rule-schemata used to generate the NoWo-PCFG. Root is the unique start-symbol, M is the set of all MRs present in the corpus, C is set the of all context-identifiers present in the corpus, T is the set of terminals of the MG, V is the vocabulary of the corpus. $Pred0(M)$ is the subset of all MRs in M of the form predicate, $Pred1(M)$ is the subset of all MRs in M of the form predicate(arg1) and $Pred2(M)$ is the subset of all MRs in M of the form predicate(arg1,arg2). $p(m)$ is the predicate of the MR m, $a1(m)$ is the first argument of the MR m, $a2(m)$ is the second argument of the MR m. The rules expanding $Phrase_x$ ensure that it contains at least one $Word_x$. A set on the right-hand side of a rule is shorthand for all possible orderings of the elements of the set.

the automatic assignment of zero probability to rules not used during training.[9]

For parsing, the resulting PCFG is slightly modified by removing the context-identifiers. This is done because the task of a semantic parser is to establish a mapping between NLs and MRs, irrespective of contexts which were only used for learning the parser and should not play a role in its final performance. To do this, we add up the probability of all rules which differ only in the context-identifier which can be thought of as marginalizing out the different contexts, giving our first model which we call NoWo-PCFG.[10]

Note that the context-deletion (and the simple smoothing) enables NoWo-PCFG to parse sentences into meanings not present in the data it was trained on which, in fact, happens. For example, there are 81 meanings in the training data for the first English match that are not present in any of the other games' training data. The PCFG trained on games 2, 3 and 4 is still able to correctly assign 12 of those 81 meanings which it has not seen during the training phase which shows the effectiveness of the bottom-up constraint.

For evaluation, we employ 4-fold cross validation as described in detail in Chen and Mooney (2008) and used by KM: the model is trained on all possible combinations of 3 of the 4 games and is then used to produce an MR for all sentences of the held-out game *for which there is a matching gold-standard meaning*. For an NL w, our model produces an MR m by finding the *most probable* parse of w with the CKY algorithm and reading off m at the S_m-node.[11] An MR is considered correct if and only if it matches the gold-standard MR *exactly*; the final evaluation result is averaged over all 4 folds. Our evaluation results for NoWo-PCFG are given in Table 2. All scores are reported in F-measure which is the harmonic mean of Precision and Recall. In this specific case, precision is the fraction of correct parses out

[9] We experimented with α=0.1, α=0.5 and α=1.0 and found that overall, 0.1 yields the best results. We also tried jittering the initial rule weights during training and found that our results are very robust and seem to be independent of a specific initialization.

[10] NoWo because this model, unlike the one described in Section 4, does **no**t make explicit use of **w**ord **o**rder generalisations.

[11] For parsing, we use Mark Johnson's freely available CKY implementation which can be downloaded at `http://web.science.mq.edu.au/~mjohnson/Software.htm`.

	English	Korean
KM	0.742	0.764
KM 'supervised'	0.810	0.808
Chen et al. (2010)	0.801	0.812
NoWo-PCFG	0.742	0.718
WO-PCFG	**0.860**	**0.829**

Table 2: A summary of results for the parsing task, in F-measure. We also show the results of Chen et al. (2010), as given in Kim and Mooney (2010), which to our knowledge are the highest previously reported scores for Korean. WO-PCFG, described in Section 4 performs better than all previously reported models, but only slightly so for Korean.

of the total number of parses the model returns. Recall is the fraction of correct parses out of the total number of test sentences.[12]

NoWo-PCFG performs a little worse than KM's model. Its scores are virtually identical for English (0.742) and worse for Korean (0.718 vs 0.764). We are not sure as to why our model performs worse on the Korean data, but it might have to do with the fact that the Korean average ambiguity is higher than for the English data.

This shows that it is not only possible to reduce the task of learning a semantic parser to standard grammatical inference, but that this way of approaching the problem yields comparable results.

The remainder of the paper focuses on our second main point: that letting the model learn additional kinds of information, such as the language's canonical word order, can further improve its performance. In order to do this we propose a model that learns the word order as well as the mapping from NLs to MRs, and compare its performance to that of the other models.

4 Extending NoWo-PCFG to WO-PCFG

We already pointed out that our model considers every possible linear order of syntactic constituents. Our NoWo-PCFG model considers each of the possible word orders for every meaning and context in isolation: it is unable to infer from the fact that most meanings it has observed are most likely to be expressed with a certain word order that new meanings it will encounter are also more likely to be expressed with this word order. It seems, however, to be at least a soft fact about languages that they *do* have a canonical word order that is more likely to be realized in its sentences than any other possible word order. In order to test whether trying to learn this order helps our model, we modify the CFG used for NoWo-PCFG so it can learn word order generalizations, and train it in the same way to yield another semantic parser, WO-PCFG.

4.1 Setting up WO-PCFG

For every possible ordering of the constituents corresponding to an MR, our grammar contains a rule. In NoWo-PCFG, these different rules all share the same parent which prevents the model from learning the probability of the different word orders corresponding to the many rules. A straight-forward way to overcome this is to annotate every S_m node with the word order of its daughter. We split every S_m non-terminal in multiple S_{wo_m} non-terminals, where $wo \in$ {v,sv,vs,svo,sov,osv,ovs,vso,vos} indicates the linear order of the constituents the non-terminal rewrites as.[13]

This in itself does not yet allow our model to *use* word order as a means of generalization. To model that whenever it *encounters a specific example* that is indicative of a certain word order, this word order becomes slightly more probable *for every other example as well*, we have to make a further slight change to the CFG which we now describe. A formally explicit description of the necessary changes which we go on to describe is given in Figure 4.

We introduce six new non-terminals, corresponding to the six possible word orders SVO, SOV, VSO, VOS, OSV and OVS and require every S_{wo_m} non-terminal to be dominated by the non-terminal compatible with its daughters linear order. As an example, consider the two syntactic non-terminals corresponding to the MR kick(pink1), $S_{vs\_kick(pink11)}$ and $S_{sv\_kick(pink11)}$. Whenever an example is successfully analyzed as $S_{vs\_kick(pink11)}$, this should strengthen our model's expectation of encountering

[12] Because our model parses every sentence, for it Recall and Precision are identical and F-measure is identical to Accuracy.

[13] We assume, somewhat simplifying, that an MR's predicate corresponds to a V(erb), its first argument corresponds to the S(ubject) and its second argument corresponds to the O(bject). These are purely formal categories that are not constrained to correspond to specific linguistic categories.

$\text{Root} \rightarrow wo$	$wo \in WO$
$wo \rightarrow S_{x\_m}$	$wo \in WO, x \in WOS, x \subset wo, m \in M$
$S_{v\_m} \rightarrow c\, Phrase_{p(m)}$	$c \in C, m \in c, m \in Pred0(M)$
$S_{x\_m} \rightarrow c\, \{Phrase_{p(m)}, Phrase_{a1(m)}\}$	$c \in C, m \in c, m \in Pred1(M), x \in \{sv, vs\}$
$S_{x\_m} \rightarrow c\, \{Phrase_{p(m)}, Phrase_{a1(m)}, Phrase_{a2(m)}\}$	$c \in C, m \in c, m \in Pred2(M), x \in WOS$
$S_{v\_\emptyset} \rightarrow c\, Phrase_{\emptyset}$	$c \in C$

Figure 4: In order to turn NoWo-PCFG described in Figure 3 into the WO-PCFG described in the text, substitute the first five rule-schemata with the schemata given here. WO is the set of word order non-terminals $\{SVO, SOV, OSV, OVS, VSO, VOS\}$, WOS is the set of word order annotations $\{v, sv, vs, svo, svo, ovs, osv, vso, vos\}$. We take $x \subset wo$ to mean that x is compatible with wo, where v is compatible with all word orders, sv is compatible with SVO,SOV and OSV, and so on. For rule-schemata 4 and 5, the choice of x determines the order of the elements of the set on the right-hand side. All other symbols have the same meaning as explained in Figure 3.

more examples where the verb precedes the subject, i.e. of the language being pre-dominantly VSO, VOS or OVS. Therefore, we allow *VSO*, *VOS* and *OVS* to be rewritten as $S_{vs\_kick(pink11)}$. More generally, every word order non-terminal can rewrite as any of the S_{wo_m} non-terminals that are compatible with it. Adding this additional layer of word order abstraction leads to a grammar with 36,019 rules for English and a grammar with 33,715 rules for Korean.

4.2 Evaluation of WO-PCFG

Training and evaluating WO-PCFG in exactly the same way as the previous grammar gives an F-measure of 0.860 for English and an F-measure of 0.829 for Korean. Those scores are, to our knowledge, the highest scores previously reported for this parsing task and establish our second main point: letting the model learn the language's word order in addition to learning the mapping from sentences to MR increases semantic parsing accuracy.[14]

An intuitive explanation for the increase in performance is that by allowing the model to learn word order, we are providing it with a new dimension along which it can generalize.

In this sense, we can look at our refinement as providing the model with abstract linguistic knowledge, namely that languages tend to have a canonical word order. The usefulness of this kind of information is impressive – for English, it improves the accuracy of semantic parsing by almost 12% in F-measure and for Korean by 11.1%. In addition, our model correctly learns that English's predominant word order is SVO and that Korean is predominantly SOV, assigning by far the highest probability to the corresponding *Root* rewrite rule (0.91 for English and 0.98 for Korean). This kind of information is useful in its own right and could, for example, be exploited by coupling word order with other linguistic properties, perhaps following Greenberg (1966)'s implicational universals.

In this sense, the reduction of grounded learning problems to grammatical inference does not only make possible the application of a wide variety of tools and insights developed over years of research, it might also make it easier to bring abstract (and not so abstract) linguistic knowledge to bear on those tasks.

The overall slightly worse performance of our system on Korean data might stem from the fact that Korean, unlike English, has a rich morphology, and that our model does not learn anything about morphology at all. We plan on further investigating effects like this in the future, as well as applying more advanced grammatical inference algorithms.

5 Conclusion and Future Work

We have shown that certain grounded learning tasks such as learning a semantic parser from semantically enriched training data can be reduced to a grammatical inference problem over strings. This allows

[14] Liang et al. (2009)'s model can be seen as capturing something similar to our word order generalization with the help of a Field Choice Model which primarily captures discourse coherence and salience properties. It differs, however, in that it can only learn one generalization for each predicate type and no language wide generalization.

for the application of techniques and insights developed for grammatical inference to grounded learning tasks. In addition, we have shown that letting the model learn the language's canonical word order improves parsing performance, beyond the top scores previously reported, thus illustrating the usefullnes of linguistic knowledge for tasks like this.

In future research, we plan to address the limitation of our model to a finite set of meaning representations, in particular through the use of non-parametric Bayesian models such as the Infinite PCFG model of Liang et al. (2007) and the Infinite Tree model of Finkel et al. (2007); both allow for a potentially infinite set of non-terminals, hence directly addressing this problem. In addition, we are thinking about using an extension of the PCFG formalism that allows for some kind of 'feature-passing' which could lead to much smaller and more general grammars.

References

N. C. Chang and T. V. Maia. 2001. Grounded learning of grammatical constructions. In *2001 AAAI Spring Symposium on Learning Grounded Representations*.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37:397–435.

Jenny R. Finkel, Trond Grenager, and Christopher D. Manning. 2007. The infinite tree. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 272–279.

Joseph H. Greenberg. 1966. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph H. Greenberg, editor, *Universals of Language*, chapter 5, pages 73–113. The MIT Press, Cambridge, Massachusetts.

Mark Johnson, Katherine Demuth, Michael Frank, and Bevan Jones. 2010. Synergies in learning words and their referents. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1018–1026.

Joohyun Kim and Raymond J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*.

K. Lari and S.J. Young. 1990. The estimation of Stochastic Context-Free Grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4(35-56).

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 783–792.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI 2005*.

Relation Extraction with Relation Topics

Chang Wang **James Fan** **Aditya Kalyanpur** **David Gondek**
IBM T. J. Watson Research Lab
19 Skyline Drive, Hawthorne, New York 10532
{wangchan, fanj, adityakal, dgondek}@us.ibm.com

Abstract

This paper describes a novel approach to the semantic relation detection problem. Instead of relying only on the training instances for a new relation, we leverage the knowledge learned from previously trained relation detectors. Specifically, we detect a new semantic relation by projecting the new relation's training instances onto a lower dimension topic space constructed from existing relation detectors through a three step process. First, we construct a large relation repository of more than 7,000 relations from Wikipedia. Second, we construct a set of non-redundant relation topics defined at multiple scales from the relation repository to characterize the existing relations. Similar to the topics defined over words, each relation topic is an interpretable multinomial distribution over the existing relations. Third, we integrate the relation topics in a kernel function, and use it together with SVM to construct detectors for new relations. The experimental results on Wikipedia and ACE data have confirmed that background-knowledge-based topics generated from the Wikipedia relation repository can significantly improve the performance over the state-of-the-art relation detection approaches.

1 Introduction

Detecting semantic relations in text is very useful in both information retrieval and question answering because it enables knowledge bases to be leveraged to score passages and retrieve candidate answers. To extract semantic relations from text, three types of approaches have been applied. Rule-based methods (Miller et al., 2000) employ a number of linguistic rules to capture relation patterns. Feature-based methods (Kambhatla, 2004; Zhao and Grishman, 2005) transform relation instances into a large amount of linguistic features like lexical, syntactic and semantic features, and capture the similarity between these feature vectors. Recent results mainly rely on kernel-based approaches. Many of them focus on using tree kernels to learn parse tree structure related features (Collins and Duffy, 2001; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005). Other researchers study how different approaches can be combined to improve the extraction performance. For example, by combining tree kernels and convolution string kernels, (Zhang et al., 2006) achieved the state of the art performance on ACE (ACE, 2004), which is a benchmark dataset for relation extraction.

Although a large set of relations have been identified, adapting the knowledge extracted from these relations for new semantic relations is still a challenging task. Most of the work on domain adaptation of relation detection has focused on how to create detectors from ground up with as little training data as possible through techniques such as bootstrapping (Etzioni et al., 2005). We take a different approach, focusing on how the knowledge extracted from the existing relations can be reused to help build detectors for new relations. We believe by reusing knowledge one can build a more cost effective relation detector, but there are several challenges associated with reusing knowledge.

The first challenge to address in this approach is how to construct a relation repository that has suffi-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1426–1436,
Edinburgh, Scotland, UK, July 27–31, 2011.

cient coverage. In this paper, we introduce a method that automatically extracts the knowledge characterizing more than 7,000 relations from Wikipedia. Wikipedia is comprehensive, containing a diverse body of content with significant depth and grows rapidly. Wikipedia's infoboxes are particularly interesting for relation extraction. They are short, manually-created, and often have a relational summary of an article: a set of attribute/value pairs describing the article's subject.

Another challenge is how to deal with overlap of relations in the repository. For example, Wikipedia authors may make up a name when a new relation is needed without checking if a similar relation has already been created. This leads to relation duplication. We refine the relation repository based on an unsupervised multiscale analysis of the correlations between existing relations. This method is parameter free, and able to produce a set of non-redundant *relation topics* defined at multiple scales. Similar to the topics defined over words (Blei et al., 2003), we define relation topics as multinomial distributions over the existing relations. The relation topics extracted in our approach are interpretable, orthonormal to each other, and can be used as basis relations to re-represent the new relation instances.

The third challenge is how to use the relation topics for a relation detector. We map relation instances in the new domains to the relation topic space, resulting in a set of new features characterizing the relationship between the relation instances and existing relations. By doing so, background knowledge from the existing relations can be introduced into the new relations, which overcomes the limitations of the existing approaches when the training data is not sufficient. Our work fits in to a class of relation extraction research based on "distant supervision", which studies how knowledge and resources external to the target domain can be used to improve relation extraction. (Mintz et al., 2009; Jiang, 2009; Chan and Roth, 2010). One distinction between our approach and other existing approaches is that we represent the knowledge from distant supervision using automatically constructed topics. When we test on new instances, we do not need to search against the knowledge base. In addition, our topics also model the indirect relationship between relations. Such information cannot be directly found from the knowledge base.

The contributions of this paper are three-fold. Firstly, we extract a large amount of training data for more than 7,000 semantic relations from Wikipedia (Wikipedia, 2011) and DBpedia (Auer et al., 2007). A key part of this step is how we handle noisy data with little human effort. Secondly, we present an unsupervised way to construct a set of *relation topics* at multiple scales. This step is parameter free, and results in a non-redundant, multiscale relation topic space. Thirdly, we design a new kernel for relation detection by integrating the relation topics into the relation detector construction. The experimental results on Wikipedia and ACE data (ACE, 2004) have confirmed that background-knowledge-based features generated from the Wikipedia relation repository can significantly improve the performance over the state-of-the-art relation detection approaches.

2 Extracting Relations from Wikipedia

Our training data is from two parts: relation instances from DBpedia (extracted from Wikipedia infoboxes), and sentences describing the relations from the corresponding Wikipedia pages.

2.1 Collecting the Training Data

Since our relations correspond to Wikipedia infobox properties, we use an approach similar to that described in (Hoffmann et al., 2010) to collect positive training data instances. We assume that a Wikipedia page containing a particular infobox property is likely to express the same relation in the text of the page. We further assume that the relation is most likely expressed in the first sentence on the page which mentions the arguments of the relation. For example, the Wikipedia page for "Albert Einstein" contains an infobox property "alma mater" with value "University of Zurich", and the first sentence mentioning the arguments is the following: "Einstein was awarded a PhD by the University of Zurich", which expresses the relation. When looking for relation arguments on the page, we go beyond (sub)string matching, and use link information to match entities which may have different surface forms. Using this technique, we are able to collect a large amount of positive training instances of DBpe-

dia relations.

To get precise type information for the arguments of a DBpedia relation, we use the DBpedia knowledge base (Auer et al., 2007) and the associated YAGO type system (Suchanek et al., 2007). Note that for every Wikipedia page, there is a corresponding DBpedia entry which has captured the infobox-properties as RDF triples. Some of the triples include type information, where the subject of the triple is a Wikipedia entity, and the object is a YAGO type for the entity. For example, the DBpedia entry for the entity "Albert Einstein" includes YAGO types such as Scientist, Philosopher, Violinist etc. These YAGO types are also linked to appropriate WordNet concepts, providing for accurate sense disambiguation. Thus, for any entity argument of a relation we are learning, we obtain sense-disambiguated type information (including super-types, sub-types, siblings etc.), which become useful generalization features in the relation detection model. Given a common noun, we can also retrieve its type information by checking against WordNet (Fellbaum, 1998).

2.2 Extracting Rules from the Training Data

We use a set of rules together with their popularities (occurrence count) to characterize a relation. A rule representing the relations between two arguments has five components (ordered): argument_1 type, argument_2 type, noun, preposition and verb. A rule example of *ActiveYearsEndDate* relation (about the year that a person retired) is:

person100007846|year115203791|-|in|retire.

In this example, argument_1 type is *person100007846*, argument_2 type is *year115203791*, both of which are from YAGO type system. The key words connecting these two arguments are *in* (preposition) and *retire* (verb). This rule does not have a noun, so we use a '-' to take the position of noun. The same relation can be represented in many different ways. Another rule example characterizing the same relation is

person100007846|year115203791|retirement|-|announce.

This paper only considers three types of words: *noun*, *verb* and *preposition*. It is straightforward to expand or simplify the rules by including more or removing some word types. The keywords are extracted from the shortest path on the dependency tree between the two arguments. A dependency tree (Figure 1) represents grammatical relations between words in a sentence. We used a slot grammar parser (McCord, 1995) to generate the parse tree of each sentence. Note that there could be multiple paths between two arguments in the tree. We only take the shortest path into consideration. The popularity value corresponding to each rule represents how many times this rule applies to the given relation in the given data. Multiple rules can be constructed from one relation instance, if multiple argument types are associated with the instance, or multiple nouns, prepositions or verbs are in the dependency path.

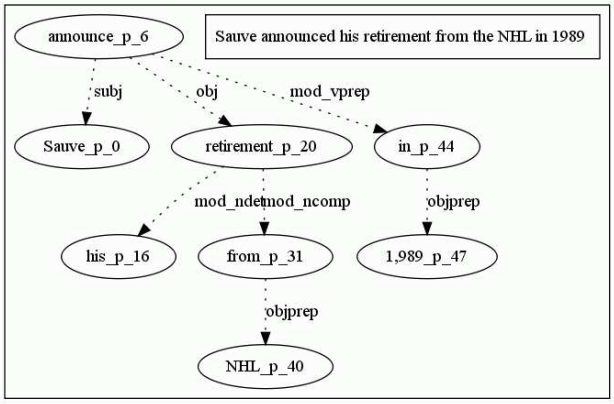


Figure 1: A dependency tree example.

2.3 Cleaning the Training Data

To find a sentence on the Wikipedia page that is likely to express a relation in its infobox, we consider the first sentence on the page that mentions both arguments of the relation. This heuristic approach returns reasonably good results, but brings in about 20% noise in the form of false positives, which is a concern when building an accurate statistical relation detector. To address this issue, we have developed a two-step technique to automatically remove some of the noisy data. In the first step, we extract popular argument types and keywords for each DBpedia relation from the given data, and then use the combinations of those types and words to create initial rules. Many of the argument types and keywords introduced by the noisy data are often not very popular, so they can be filtered out in the first step. Not all initial rules make sense. In the second step, we

check each rule against the training data to see if that rule really exists in the training data or not. If it does not exist, we filter it out. If a sentence does not have a single rule passing the above procedure, that sentence will be removed. Using the above techniques, we collect examples characterizing 7,628 DBpedia relations.

3 Learning Multiscale Relation Topics

An extra step extracting knowledge from the raw data is needed for two reasons: Firstly, many DBpedia relations are inter-related. For example, some DBpedia relations have a subclass relationship, e.g. "AcademyAward" and "Award"; others overlap in their scope and use, e.g., "Composer" and "Artist"; while some are equivalent, e.g., "DateOfBirth" and "BirthDate". Secondly, a fairly large amount of the noisy labels are still in the training data.

To reveal the intrinsic structure of the current DBpedia relation space and filter out noise, we carried out a correlation analysis of relations in the training data, resulting in a relation topic space. Each relation topic is a multinomial distribution over the existing relations. We adapted diffusion wavelets (Coifman and Maggioni, 2006) for this task. Compared to the other well-known topic extraction methods like LDA (Blei et al., 2003) and LSI (Deerwester et al., 1990), diffusion wavelets can efficiently extract a hierarchy of interpretable topics without any user input parameter (Wang and Mahadevan, 2009).

3.1 An Overview of Diffusion Wavelets

The diffusion wavelets algorithm constructs a compressed representation of the dyadic powers of a square matrix by representing the associated matrices at each scale not in terms of the original (unit vector) basis, but rather using a set of custom generated bases (Coifman and Maggioni, 2006). Figure 2 summarizes the procedure to generate diffusion wavelets. Given a matrix T, the $\mathcal{QR}$ (a modified QR decomposition) subroutine decomposes T into an orthogonal matrix Q and a triangular matrix R such that $T \approx QR$, where $|T_{i,k} - (QR)_{i,k}| < \varepsilon$ for any i and k. Columns in Q are orthonormal basis functions spanning the column space of T at the finest scale. RQ is the new representation of T with respect to the space spanned by the columns of Q (this result is based on the matrix invariant subspace theory). At an arbitrary level j, $\mathcal{DWT}$ learns the basis functions from T^{2^j} using $\mathcal{QR}$. Compared to the number of basis functions spanning T^{2^j}'s original column space, we usually get fewer basis functions, since some high frequency information (corresponding to the "noise" at that level) can be filtered out. $\mathcal{DWT}$ then computes $T^{2^{j+1}}$ using the low frequency representation of T^{2^j} and the procedure repeats.

$\{[\phi_j]_{\phi_0}\} = \mathcal{DWT}(T, \varepsilon, J)$
//**INPUT**:
//T: The input matrix.
//ε: Desired precision, which can be set to a small number or simply machine precision.
//J: Number of levels (optional).
//**OUTPUT**:
//$[\phi_j]_{\phi_0}$: extended diffusion scaling functions at scale j.

$\phi_0 = I$;
$For\ j = 0\ to\ J-1\ \{$
  $([\phi_{j+1}]_{\phi_j}, [T^{2^j}]_{\phi_j}^{\phi_{j+1}}) \leftarrow \mathcal{QR}([T^{2^j}]_{\phi_j}^{\phi_j}, \varepsilon)$;
  $[\phi_{j+1}]_{\phi_0} = [\phi_{j+1}]_{\phi_j}[\phi_j]_{\phi_0}$;
  $[T^{2^{j+1}}]_{\phi_{j+1}}^{\phi_{j+1}} = ([T^{2^j}]_{\phi_j}^{\phi_{j+1}}[\phi_{j+1}]_{\phi_j})^2$;
$\}$

Figure 2: Diffusion Wavelets construct multiscale representations of the input matrix at different scales. $\mathcal{QR}$ is a modified QR decomposition. J is the max step number (this is optional, since the algorithm automatically terminates when it reaches a matrix of size 1×1). The notation $[T]_{\phi_a}^{\phi_b}$ denotes matrix T whose column space is represented using basis ϕ_b at scale b, and row space is represented using basis ϕ_a at scale a. The notation $[\phi_b]_{\phi_a}$ denotes basis ϕ_b represented on the basis ϕ_a. At an arbitrary scale j, we have p_j basis functions, and length of each function is l_j. The number of p_j is determined by the intrinsic structure of the given dataset in $\mathcal{QR}$ routine. $[T]_{\phi_a}^{\phi_b}$ is a $p_b \times l_a$ matrix, and $[\phi_b]_{\phi_a}$ is an $l_a \times p_b$ matrix.

3.2 Constructing Multiscale Relation Topics

Learning Relation Correlations

Assume we have M relations, and the i^{th} of them is characterized by m_i <*rule, popularity*> pairs. We use $s(a, b)$ to represent the similarity between the a^{th} and b^{th} relations. To compute $s(a, b)$, we first normalize the popularities for each relation, and then

look for the rules that are shared by both relation a and b. We use the product of corresponding popularity values to represent the similarity score between two relations with respect to each common rule. $s(a, b)$ is set to the sum of such scores over all common rules. The relation-relation correlation matrix S is constructed as follows:

$$S = \begin{bmatrix} s(1,1) & \cdots & s(1,M) \\ \cdots & \cdots & \cdots \\ s(M,1) & \cdots & s(M,M) \end{bmatrix}$$

We have more than $200,000$ argument types, tens of thousands of distinct nouns, prepositions, and verbs, so we potentially have trillions of distinct rules. One rule may appear in multiple relations. The more rules two relations share, the more related two relations should be. The rules shared across different relations offer us a novel way to model the correlations between different relations, and further allow us to create relation topics. The rules can also be simplified. For example, we may treat argument$_1$, argument$_2$, noun, preposition and verb separately. This results in simple rules that only involve in one argument type or word. The correlations between relations are then computed only based on one particular component like argument$_1$, noun, etc.

Theoretical Analysis

Matrix S models the correlations between relations in the training data. Once S is constructed, we adapt diffusion wavelets (Coifman and Maggioni, 2006) to automatically extract the basis functions spanning the original column space of S at multiple scales. The key strength of the approach is that it is data-driven, largely parameter-free and can automatically determine the number of levels of the topical hierarchy, as well as the topics at each level. However, to apply diffusion wavelets to S, we first need to show that S is a positive semi-definite matrix. This property guarantees that all eigenvalues of S are ≥ 0. Depending on the way we formalize the rules, the methods to validate this property are slightly different. When we treat argument$_1$, argument$_2$, noun, preposition and verb separately, it is straightforward to see the property holds. In Theorem 1, we show the property also holds when we use more complicated rules (using the 5-tuple rule in Section 2.2 as an example in the proof).

Theorem 1. *S is a Positive Semi-Definite matrix.*

Proof: An arbitrary rule r_i is uniquely characterized by a five tuple: *argument$_1$ type*| *argument$_2$ type*| *noun*| *preposition*| *verb*. Since the number of distinct argument types and words are constants, the number of all possible rules is also a constant: R.

If we treat each rule as a feature, then the set of rules characterizing an arbitrary relation r_i can be represented as a point $[p_i^1, \cdots, p_i^R]$ in a latent R dimensional rule space, where p_i^j represents the popularity of rule j in relation r_i in the given data.

We can verify that the way to compute $s(a, b)$ is the same as $s(a, b) =< [p_a^1 \cdots p_a^R], [p_b^1 \cdots p_b^R] >$, where $< \cdot, \cdot >$ is the cosine similarity (kernel). It follows directly from the definition of positive semi-definite matrix (PSD) that S is PSD (Schölkopf and Smola, 2002). □

In our approach, we construct multiscale relation topics by applying $\mathcal{DWT}$ to decompose $S/\lambda_{max}(S)$, where $\lambda_{max}(S)$ represents the largest eigenvalue of S. Theorem 2 shows that this decomposition will converge, resulting in a relation topic hierarchy with one single topic at the top level.

Theorem 2. *Let $\lambda_{max}(S)$ represent the largest eigenvalue of matrix S, then $\mathcal{DWT}(S/\lambda_{max}(S), \varepsilon)$ produces a set of nested subspaces of the column space of S, and the highest level of the resulting subspace hierarchy is spanned by one basis function.*

Proof: From Theorem 1, we know that S is a PSD matrix. This means $\lambda_{max}(S) \in [0, +\infty)$ (all eigenvalues of S are non-negative). This further implies that $\Lambda(S)/\lambda_{max}(S) \in [0, 1]$, where $\Lambda(S)$ represents any eigenvalue of S.

The idea underlying diffusion wavelets is based on decomposing the spectrum of an input matrix into various spectral bands, spanned by basis functions (Coifman and Maggioni, 2006). Let $T = S/\lambda_{max}(S)$. In Figure 2, we construct spectral bands of eigenvalues, whose associated eigenvectors span the corresponding subspaces. Define dyadic spatial scales t_j as

$$t_j = \sum_{t=0}^{j} 2^t = 2^{j+1} - 1, \;\; j \geq 0 \, .$$

At each spatial scale, the spectral band is defined as:

$$\Omega_j(T) = \{\lambda \in \Lambda(T), \lambda^{t_j} \geq \varepsilon\},$$

where $\Lambda(T)$ represents any eigenvalue of T, and $\varepsilon \in (0, 1)$ is a pre-defined threshold in Figure 2. We can now associate with each of the spectral bands a vector subspace spanned by the corresponding eigenvectors:

$$V_j = \langle \{\xi_\lambda : \lambda \in \Lambda(T), \lambda^{t_j} \geq \varepsilon\} \rangle, \;\; j \geq 0 \, .$$

In the limit, we obtain

$$\lim_{j \to \infty} V_j = \langle \{\xi_\lambda : \lambda = 1\} \rangle$$

That is, the highest level of the resulting subspace hierarchy is spanned by the eigenvector associated with the largest eigenvalue of T. □

This result shows that the multiscale analysis of the relation space will automatically terminate at the level spanned by one basis, which is the most popular relation topic in the training data.

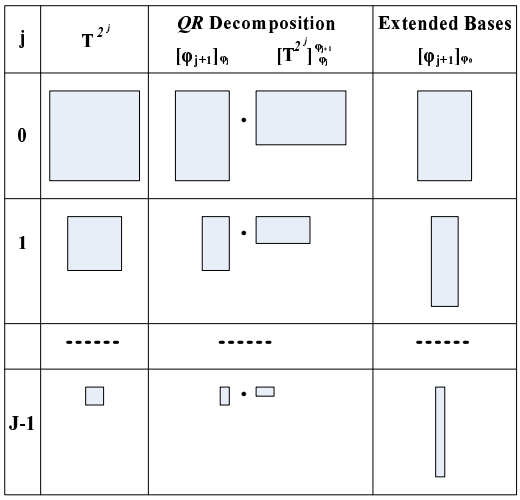


Figure 3: Learning Relation Topics at Multiple Scales.

3.3 High Level Explanation

We first create a set of rules to characterize each input relation. Since these rules may occur in multiple relations, they provide a way to model the co-occurrence relationship between different relations. Our algorithm starts with the relation co-occurrence matrix and then repeatedly applies QR decomposition to learn the topics at the current level while at the same time modifying the matrix to focus more on low-frequency indirect co-occurrences (between relations) for the next level. Running $\mathcal{DWT}$ is equivalent to running a Markov chain on the input data forward in time, integrating the local geometry and therefore revealing the relevant geometric structures of the whole data set at different scales. At scale j, the representation of $T^{2^{j+1}}$ is compressed based on the amount of remaining information and the desired precision. This procedure is illustrated in Figure 3. In the resulting topic space, instances with related relations will be grouped together. This approach may significantly help us detect new relations, since it potentially expands the information brought in by new relation instances from making use of the knowledge extracted from the existing relation repository.

3.4 Benefits

As shown in Figure 3, the topic spaces at different levels are spanned by a different number of basis functions. These numbers reveal the dimensions of the relevant geometric structures of data at different levels. These numbers are completely data-driven: the diffusion wavelets approach can automatically find the number of levels and simultaneously generate the topics at each level. Experiments show that most multiscale topics are interpretable (due to the sparsity of the scaling functions), such that we can interpret the topics at different scales and select the best scale for embedding. Compared to bootstrapping approach, our approach is accumulative; that is as the system learns more relations, it gets better at learning new relations. Because our approach takes advantage of the previously learned relations, and the topic space is enriched as we learn more and more relations.

We use diffusion wavelets (DWT) rather than other hierarchy topic models like hLDA (Blei et al., 2004) to extract relation topics for two reasons. First, DWT is parameter free while other models need some user-input parameters like hierarchy level. Second, DWT is more efficient than the other models. After the relation correlation matrix is constructed, DWT only needs a couple of minutes to extract multiscale topics on a regular computer. A direct experimental comparison between DWT and hLDA can be found in (Wang and Mahadevan, 2009).

4 Constructing Relation Detectors with Multiscale Relation Topics

4.1 Project Relation Instances onto Topics

When we design detectors for new relations, we treat arg_1, arg_2, $noun$, and $verb$ separately to get stronger correlations between relations. We do not directly use $preposition$. Any DBpedia relation $r \in \{1, \cdots, M\}$ is represented with 4 vectors $r_t = [r_t(1), \cdots, r_t(N_t)]$, where $t \in \{arg_1, arg_2, noun, verb\}$, N_t represents the size of the vocabulary set of the type t component in the Wikipedia training data, and $r_t(j)$ represents the occurrence count of type t component in relation r. For example, N_{verb} is the size of the verb vocabulary set in the training data and $r_{verb}(j)$ represents the occurrence count of the j^{th} verb in relation r. When a new relation instance x is given, we extract the dependency path between two arguments, and create four vectors x_t, where $t \in \{arg_1, arg_2, noun, verb\}$, following the same format as r_t. The projection result of x_t onto the DBpedia relation space X_t is as follows:

$X_t = [< r_t(1), x_t(1) >, \cdots, < r_t(M), x_t(M) >]$, where $< \cdot, \cdot >$ is the cosine similarity of two vectors. At level k, the embedding of x is $E^k_x = [E^k_{X_{arg_1}}, E^k_{X_{arg_2}}, E^k_{X_{noun}}, E^k_{X_{verb}}]$, where $E^k_{X_t} = ([\phi_k]_{\phi_0})^T X_t$, and $[\phi_k]_{\phi_0}$ is defined in Figure 2.

4.2 Design New Kernel Using Topic Features

We combine E^k_x with 3 existing kernels ($K_{Argument}$, K_{Path} and K_{BOW}) to create a new kernel for relation detection.

(1) $K_{Argument}$ matches two arguments, it returns the number of common argument types that the input arguments share.

(2) K_{Path} matches two dependency paths. This kernel is formally defined in (Zhao and Grishman, 2005). We extended this kernel by also matching the common nouns, prepositions and verbs in the dependency paths. We assign weight 1 to verbs, 0.5 to nouns and prepositions.

(3) K_{BOW} models the number of common nouns, prepositions and verbs in the given sentences but not in the dependency paths. Since these words are not as important as the words inside the dependency path, we assign weight 0.25 to them.

(4) $K_{TF_k}(x, y) =< E^k_x, E^k_y >$, where x, y are two input relation instances, and $< \cdot, \cdot >$ models the cosine similarity of two vectors. TF stands for topic feature.

(5) The final kernel used in this paper is

$\alpha_1 K_{Argument} + \alpha_2 K_{Path} + \alpha_3 K_{BOW} + \alpha_4 K_{TF_k}$, where α_i can be tuned for each individual domain. In this paper, we set $\alpha_i = 1$ for $i \in \{1, 2, 3, 4\}$.

4.3 Algorithm to Construct Relation Detectors

1. **Construct a relation repository from Wikipedia.**
   (a) Collect training data from Wikipedia and DBpedia (Section 2.1);
   (b) Clean the data representing each input relation (Section 2.2 and 2.3);
   (c) Create relation correlation matrix S following the approach described in Section 3.2, resulting in an $M \times M$ matrix.
2. **Create multiscale relation topics.**
   $[\phi_k]_{\phi_0} = \mathcal{DWT}(S/\lambda_{max}(S), \varepsilon)$, where $\mathcal{DWT}()$ is the diffusion wavelets implementation described in Section 3.1. $[\phi_k]_{\phi_0}$ are the scaling function bases at level k represented as an $M \times p_k$ matrix, $k = 1, \cdots, h$ represents the level in the topic hierarchy. The value of p_k is determined in $\mathcal{DWT}()$ based on the intrinsic structure of the given dataset. Columns of $[\phi_k]_{\phi_0}$ are used as relation topics at level k.
3. **Construct relation detectors for new relations.**
   Given the training data from a new relation, project the data onto level k of the multiscale topic hierarchy, where k is chosen by users (Section 4.1). Apply SVM classifiers together with our kernel (Section 4.2) to create detectors for new relations.

5 Experimental Results

We used SVMLight (Joachims, 1999) together with the user defined kernel setting in our approach. The trade-off parameter between training error and margin c is 1 for all experiments. Our approach to learn multiscale relation topics is largely parameter free. The only parameter to be set is the precision $\varepsilon = 10^{-5}$, which is also the default value in the diffusion wavelets implementation.

5.1 Learning Multiscale Relation Topics

Following the approach discussed in Section 2.1, we collect more than 620,000 training instances for

Table 1: Number of topics at different levels (DBpedia Relations) under 5 different settings: use args, noun, preposition and verb; arg_1 only; arg_2 only; noun only and verb only.

Level	args & words	arg_1	arg_2	noun	verb
1	7628	7628	7628	7628	7628
2	269	119	155	249	210
3	32	17	19	25	35
4	7	5	5	7	10
5	3	2	3	4	4
6	2	1	2	2	2
7	1		1	1	1

7,628 DBpedia relations. For any given topic vector v, we know it is a column vector of length M, where M is the size of the DBpedia relation set and $\|v\|$ = 1. The entry $v[i]$ represents the contribution of relation i to this topic. To explain the main concept of topic v, we sort the entries on v and print out the relations corresponding to the top entries. These relations summarize the topics in the relation repository. One topic example is as follows: [*doctoraladvisor (0.683366), doctoralstudents (0.113201), candidate (0.014662), academicadvisors (0.008623), notablestudents (0.003829), college (0.003021), operatingsystem (0.002964), combatant (0.002826), influences (0.002285), training (0.002148),* $\cdots$], where $doctoraladvisor$ is a DBpedia relation and 0.683366 is its contribution to the topic. The length of this relation vector is 7,628. We only list the top 10 relations here.

Our approach identifies 5 different topic hierarchies under different settings (use args, noun, preposition and verb; arg_1 only; arg_2 only; noun only and verb only). The number of the topics at each level is shown in Table 1. At the first level, each input relation is treated as a topic. At the second level, numbers of topics go down to reasonable numbers like 269. Finally at the top level, the number of topic is down to 1 (Theorem 2 also proves this). We show some topic examples under the first setting. The 3 topics at level 5 are shown in Table 2. They represent the most popular DBpedia relation topics. Almost all 269 topics at level 5 look semantically meaningful. They nicely capture the related relations. Some examples are in Table 3.

Table 2: 3 topics at level 5 (all word types and args).

Top 4 Relations and Their Contributions
starring 86.6%, writer 3.8%, producer 3.2%, director 1.6%
birthplace 75.3%, clubs 6.1%, deathplace 5.1%, location 4.1%
clubs 55.3%, teams 9.3%, nationalteam 6.3% college 6.0%

Table 3: Some topics at level 2 (all word types and args).

Top Relations
activeyearsenddate, careerend, finalyear, retired
commands, partof, battles, notablecommanders
occupation, shortdescription, profession, dates
influenced, schooltradition, notableideas, maininterests
destinations, end, through, posttown
prizes, award, academyawards, highlights
inflow, outflow, length, maxdepth
after, successor, endingterminus
college, almamater, education

5.2 Relation Detection on Wikipedia Data

In previous experiment, 20,000 relation instances were held and not used to construct the topic space. These instances are randomly selected from 100 relations (200 instances from each relation). This set is used as a benchmark to compare different relation detection approaches. In this experiment, 100 instances from each relation are used for training, and the other 100 are for testing. In training, we try three different settings: $n = 5$, 20 and 100, where n is the size of the training set for each relation. When we train a model for one relation, we use the training positive instances from the other 99 relations as training negatives. For example, we use 5 training positive instances and 5*99=495 training negatives to train a detector for each relation.

We compare our approach against the regular rule-based approach (Lin and Pantel, 2001) and two other kernel-based approaches (presented in Section 4.2) for relation detection task. The comparison results are summarized in Table 4. The approach using relation topics (level 2) consistently outperforms the other three approaches in all three settings. When $n = 5$, it achieves the largest improvement over the other three. This indicates that using relation topics that integrate the knowledge extracted from the existing relations, can significantly benefit us when the training data is insufficient. This is reasonable, since the prior knowledge becomes more valuable in this scenario.

The users can select the level that is the most appropriate for their applications. In this example, we only have alignment results at 7 levels. Choosing the space at level 2 spanned by a couple of hundreds of basis functions is a natural choice, since the levels below and above this have too many or too few features, respectively. A user can also select the most appropriate level by checking if the related relation topics are meaningful for their applications.

5.3 Relation Detection on ACE Data

In this experiment, we use the news domain documents of the ACE 2004 corpus (ACE, 2004) to compare our approaches against the state-of-the-art approaches. This dataset includes 348 documents and around 4400 relation instances. 7 relation types, 7 entity types, numerous relation sub-types, entity sub-types, and mention types are defined on this set. The task is to classify the relation instances into one of the 7 relation types or "NONE", which means there is no relation. For comparison, we use the same setting as (Zhang et al., 2006), by applying a 5-fold cross-validation. The scores reported here are the average of all 5 folds. This is also how the other approaches are evaluated. In this test, we treat entity types, entity sub-types and mention types equally as argument types. Table 5 summarizes the performance after applying the kernels presented in Section 4.2 incrementally, showing the improvement from each individual kernel. We also compare our approaches to the other state-of-the-art approaches including Convolution Tree kernel (Collins and Duffy, 2001), Syntactic kernel (Zhao and Grishman, 2005), Composite kernel (linear) (Zhang et al., 2006) and the best kernel in (Nguyen et al., 2009). Our approach with relation topics at level 2 has the best performance, achieving a 73.24% F-measure. The impact of the relation topics is huge. They improve the F-measure from 61.15% to 73.24%. We also test our approach using the topics at level 3. The performance is slightly worse than using level 2, but still better than the others.

This paper studies how relation topics extracted from Wikipedia relation repository can help improve relation detection performance. We do not want to tune our approach to one particular relation detection task, like ACE 2004. In our experiments, no parameter tuning was taken and no domain specific heuristic rules were applied. We are aware of some methods that could stack on our approach to further improve the performance on ACE test. The Composite kernel result in Table 5 is based on a linear combination of the Argument kernel and Convolution Tree kernel. (Zhang et al., 2006) showed that by carefully choosing the weight of each component and using a polynomial expansion, they could achieve the best performance on this data: 72.1% F-measure. (Nguyen et al., 2009) further showed that the performance can be improved by taking syntactic and semantic structures into consideration. They used several types of syntactic information including constituent and dependency syntactic parse trees to improve the state of the art approaches to 71.5% on F-measure. Heuristic rules extracted from the target data can also help improve the performance. (Jiang and Zhai, 2007) reported that by taking several heuristic rules they can improve the F-measure of Composite Kernel to 70.4%. They also showed that using maximum entropy classifier rather than SVM achieved the best performance on this task: 72.9% F-measure. To the best of our knowledge, the most recent result was reported by (Zhou and Zhu, 2011), who extended their previous work in (Zhou et al., 2007). By using several heuristics to define an effective portion of constituent trees, and training the classifiers using ACE relation sub-types (rather than on types), they achieved an impressive 75.8% F-measure. However, as pointed out in (Nguyen et al., 2009), such heuristics are tuned on the target relation extraction task and might not be appropriate to compare against the automatic learning approaches. Even though we have not done any domain specific parameter tuning or applied any heuristics, our approach still achieve significant improvements over all approaches mentioned above except one, which is based on heuristics extracted from the target domain. This also implies that by combining some of the above ideas with relation topics, the performance on ACE data may be further improved.

6 Conclusions

This paper proposes a novel approach to create detectors for new relations integrating the knowledge extracted from the existing relations. The contributions of this paper are three-fold. Firstly, we pro-

Table 4: F-measure comparison of different approaches over 100 DBpedia relations with 5, 20 and 100 positive examples per relation. AG: $K_{Argument}$, DP: K_{Path}, BOW: K_{BOW}, TF_k: K_{TF_k}.

Approaches	100	20	5
Rule Based	37.70%	27.45%	13.20%
AG+ DP	73.64%	51.85%	22.95%
AG+ DP+ BOW	78.74%	62.76%	31.98%
AG+ DP+ BOW+ TF_2	81.18%	68.03%	41.60%

Table 5: Performance comparison of different approaches with SVM over the ACE 2004 data. P: Precision, R: Recall, F: F-measure, AG: $K_{Argument}$, DP: K_{Path}, BOW: K_{BOW}, TF_k: K_{TF_k}.

Approaches	P(%)	R(%)	F(%)
Convolution Tree Kernel	72.5	56.7	63.6
Composite Kernel (linear)	73.50	67.00	70.10
Syntactic Kernel	69.23	70.50	69.86
Nguyen, et al. (2009)	76.60	67.00	71.50
AG	59.56	46.22	52.02
AG + DP	64.44	54.93	59.28
AG + DP + BOW	62.00	61.19	61.15
AG + DP + BOW + TF_3	69.63	76.51	72.90
AG + DP + BOW + TF_2	69.15	77.88	73.24

vide an automatic way to collect training data for more than 7,000 relations from Wikipedia and DBpedia. Secondly, we present an unsupervised way to construct a set of relation topics at multiple scales. Different from the topics defined over words, relation topics are defined over the existing relations. Thirdly, we design a new kernel for relation detection by integrating the relation topics in the representation of the relation instances. By leveraging the knowledge extracted from the Wikipedia relation repository, our approach significantly improves the performance over the state-of-the-art approaches on ACE data. This paper makes use of all DBpedia relations to create relation topics. It is possible that using a subset of them (more related to the target relations) might improve the performance. We will explore this in future work.

Acknowledgments

We thank the reviewers for their helpful comments. This material is based upon work supported in part by the IBM DeepQA (Watson) project. We also gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0172. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

ACE. 2004. The automatic content extraction projects, http://projects.ldc.upenn.edu/ace/.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference, Busan, Korea*, pages 11–15. Springer.

D. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. 2004. Hierarchical topic models and the nested Chinese restaurant process. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 152–160.

R. Coifman and M. Maggioni. 2006. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21:53–94.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 625–632.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–429.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland,

Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165:91–134.

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

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 286–295.

Jing Jiang and Chengxiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 113–120.

Jing Jiang. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1012–1020.

T. Joachims. 1999. *Making Large-Scale SVM Learning Practical*. MIT Press.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*.

Dekang Lin and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Michael McCord. 1995. Slot grammar: A system for simpler construction of practical natural language grammars. *Communications of the ACM*, 38(11).

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1003–1011.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

B. Schölkopf and A. J. Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A large ontology from Wikipedia and WordNet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217.

C. Wang and S. Mahadevan. 2009. Multiscale analysis of document corpora based on diffusion models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1592–1597.

Wikipedia. 2011. http://www.wikipedia.org/.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 419–426.

G. Zhou and Q. Zhu. 2011. Kernel-based semantic relation detection and classification via enriched parse tree structure. *Journal of Computer Science and Technology*, 26:45–56.

G. Zhou, M. Zhang, D. Ji, and Q. Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*.

Extreme Extraction -- Machine Reading in a Week

Marjorie Freedman, Lance Ramshaw, Elizabeth Boschee, Ryan Gabbard, Gary Kratkiewicz, Nicolas Ward, Ralph Weischedel
Raytheon BBN Technologies
10 Moulton St.
Cambridge, MA 02138
mfreedma,lramshaw,eboschee,rgabbard,kratkiewicz,
nward,weischedel@bbn.com

Abstract

We report on empirical results in *extreme extraction*. It is extreme in that (1) from receipt of the ontology specifying the target concepts and relations, development is limited to one week and that (2) relatively little training data is assumed. We are able to surpass human recall and achieve an F1 of 0.51 on a question-answering task with less than 50 hours of effort using a hybrid approach that mixes active learning, bootstrapping, and limited (5 hours) manual rule writing. We compare the performance of three systems: extraction with handwritten rules, bootstrapped extraction, and a combination. We show that while the recall of the handwritten rules surpasses that of the learned system, the learned system is able to improve the overall recall and F1.

1 Introduction

Throughout the Automatic Content Extraction [1] (ACE) evaluations and the Message Understanding Conferences[2] (MUC), teams typically had a year or more from release of the target to submitting system results. One exception was MUC-6 (Grishman & Sundheim, 1996), in which scenario templates for changing positions were extracted given only one month. Our goal was to confine development to a calendar week, in fact, <50 person hours. This is significant in two ways: the less effort it takes to bring up a new domain, (1) the more broadly applicable the technology is and (2) the less effort required to run a diagnostic research experiment.

Our second goal concerned minimizing training data. Rather than approximately 250k words of entity and relation annotation as in ACE, only ~20 example pairs per relation-type were provided as training. Reducing the training requirements has the same two desirable outcomes: demonstrating that the technology can be broadly applicable and reducing the overhead for running experiments.

The system achieved recall of 0.49 and precision of 0.53 (for an F1 of 0.51) on a blind test set of 60 queries of the form $R_i(arg_1, arg_2)$, where R_i is one of the 5 new relations and exactly one of arg_1 or arg_2 is a free variable for each query.

Key to this achievement was a hybrid of:

- a variant of (Miller, et al., 2004) to learn two new classes of entities via automatically induced word classes and active learning (6 hours)
- bootstrap relation learning (Freedman et al, 2010) to learn 5 new relation classes (2.5 hours),
- handwritten patterns over predicate-argument structure (5 hours), and
- coreference (20 hours)

Our bootstrap learner is initialized with relation tuples (not annotated text) and uses LDC's Gigaword and Wikipedia as a background corpus to learn patterns for relation detection that are based on normalized predicate argument structure as well as surface strings.

These early empirical results suggest the following: (1) It is possible to specify a domain, adapt our system, and complete manual scoring, includ-

[1] http://www.nist.gov/speech/tests/ace/
[2] http://www-nlpir.nist.gov/related_projects/muc/

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1437–1446,
Edinburgh, Scotland, UK, July 27–31, 2011.

ing human performance, within a month. Experiments in machine reading (and in extraction) can be performed much more quickly and cheaply than ever before. (2) Through machine learning and limited human pattern writing (6 hours), we adapted a machine reading system within a week (using less than 50 person hours), achieving question answering performance with an F1 of 0.5 and with recall 11% higher (relative) to a human reader. (3) Unfortunately, machine learning, though achieving 80% precision,[3] significantly lags behind a gifted human pattern writer in recall. Thus, bootstrap learning with much higher recall at minimal sacrifice in precision is highly desirable.

2 Related Work

This effort is evaluated extrinsically via formal questions expressed as a binary relation with one free variable. This contrasts with TREC Question Answering,[4] where the questions are in natural language, and not restricted to a single binary relation. Like the "list" queries of TREC QA, the requirement is to find all answers, not just one. Though question interpretation is not required in our work, interpretation of the text corpus is.

The goal of rapid adaptation has been tested in other contexts. In 2003, a series of experiments in adapting to a new language in less than month tested system performance on Cebuano and Hindi. The primary goal was to adapt to a new language, rather than a new domain. The extraction participants focused on named-entity recognition, not relation extraction (May, et al, 2003; Sekine & Grishman, 2003; Li & McCallum, 2003; Maynard et al, 2003). The scenario templates of MUC-6 (Grishman & Sundheim, 1996) are more similar to our relation extraction task, although the domain is quite different. Our experiment allowed for 1 week of development time, while MUC-6 allowed a month. The core entities in the MUC-6 task (people and organizations) had been worked on previously. In contrast all of our relations included at least one novel class. While MUC-6 systems tended to use finite-state patterns, they did not incorporate bootstrapping or patterns based on the output of a statistical parser.

For learning entity classes, we follow Miller, et al., (2004), using word clustering and active learning to train a perceptron model, but unlike that work we apply the technique not just to names but also to descriptions. An alternative approach to learning classes, applying structural patterns to bootstrap description recognition without active learning, is seen in Riloff (1996) and Kozareva et al., (2008)

Much research (e.g. Ramshaw 2001) has focused on learning relation extractors using large amounts of supervised training, as in ACE. The obvious weakness of such approaches is the resulting reliance on manually annotated examples, which are expensive and time-consuming to create.

Others have explored bootstrap relation learning from seed examples. Agichtein & Gravano (2000) and Ravichandran & Hovy (2002) reported results for generating surface patterns for relation identification; others have explored similar approaches (e.g. Pantel & Pennacchiotti, 2006). Mitchell et al. (2009) showed that for macro-reading, precision and recall can be improved by learning a large set of interconnected relations and concepts simultaneously. None use coreference to find training examples; all use surface (word) patterns. Freedman et. al (2010) report improved performance from using predicate structure for bootstrappped relation learning.

Most approaches to automatic pattern generation have focused on precision, e.g., Ravichandran and Hovy (2002) report results in TREC QA, where extracting one instance of a relation can be sufficient, rather than detecting all instances. Mitchell et al. (2009), while demonstrating high precision, do not measure recall.

By contrast, our work emphasizes recall, not just precision. Our question answering task asks list-like questions that require multiple answers. We also include the results of a secondary, extraction evaluation which requires that the system identify every mention of the relations in a small set of documents. This evaluation is loosely based on the relation mention detection task in ACE.

3 Task Set-Up and Evaluation

Our effort was divided into four phases. During the first phase, a third party produced an ontology and the resources, which included: brief (~1 paragraph) guidelines for each relation and class in the ontolo-

[3] Handwritten patterns achieved 52% precision.
[4] http://trec.nist.gov/data/qamain.html

gy; ~20 examples for each relation in the ontology; 2K documents that are rich in domain relations.

Table 1 lists the 5 new relations and number of examples provided for each. Arguments in italics were known by the system prior to the evaluation.

Relation	Ex.
possibleTreatment(Substance, Condition)	23
expectedDateOnMarket(Substance, *Date*)	11
responsibleForTreatment(Substance, A*gent*)	19
studiesDisease(*Agent*, Condition)	16
hasSideEffect(Substance, Condition)	27

Table 1: New Relations and Number of Examples

In phase two, we spent one week extending our extraction system for the new ontology. During the third phase, we ran our system over 10K documents to extract all instances of domain relations from those documents. In the fourth phase, our question answering system used the extracted information to answer queries.

4 Approach to Domain Specialization

Our approach to extracting domain relations integrated novel relation and class detectors into an existing extraction system, designed primarily around the ACE tasks. The existing system uses a discriminatively trained classifier to detect the entity and value types of ACE. It also produces a syntactic parse for each sentence; normalizes these parses to find logical predicate argument structure; and detects and coreferences pronominal, nominal, and name mentions for each of the 7 ACE entity types (Person, Organization, Geopolitical Entity, Location, Facility, Weapon, and Vehicle).[5]

The extraction system has three components that allow for rapid adaptation to a new domain:

- Class detectors trained using word classes derived from unsupervised clustering and sentence-selected training data.
- A bootstrap relation learner which given a few seed examples learns patterns that indicate the presence of relations.
- An expressive pattern language which allows a developer to express rules for relation extraction in a simple, but fast manner.

Component	Approach	Effort
Class Recognizer	Active Learning	6 hrs
Class Recognizer	Web-Mined List	1 hrs
Relation Recognizer	Semi-supervised Bootstrapping	8.5 hrs
Relation Recognizer	Manual Patterns	5 hrs
Coreference	Heuristics	20 hrs

Table 2: Effort and Approach for New Domain

4.1 Class Extraction

Each of the relations in the new domain included at least one argument that was new. While question answering requires the system to identify the classes only when they appear in a relation, knowledge of when a class is present provides important information for relation extraction. For example in our ontology, Y is a treatment for X only if Y is a substance. Thus, '*Group counseling sessions are effective treatments for depression'* does not contain an instance of *possibleTreatment*(), while '*SSRIs are effective treatments for depression*' does. The bootstrap learner allows constraints based on argument type. To use this capability, we trained the recognizer at the beginning of the week of domain adaptation and used the predicted classes during learning.

We annotated 1064 sentences (~31K words) using active learning combined with unsupervised word clusters (Miller,et al., 2004) for the following classes: *Substance-Name, Substance-Description, Condition-Name, and Condition-Description*. Generic noun-phrases like *new drugs, the illness*, etc were labeled as descriptors. Because of the time frame, we did not develop extensive guidelines nor measure inter-annotator agreement. Annotation took 6 hours. We supplemented our annotation with lists of substances and treatments from the web, which took 1 hour.

4.2 Coreference

Providing a name reference is generally preferable to a non-specific string (e.g. *the drugs*), but not always feasible; for instance, reports of new research may appear without a name for the drug. Our existing system's coreference algorithms operate only on mentions of ACE entity types (persons, organizations, GPEs, other locations, facilities, vehicles, and weapons). During the week of domain adaption we developed new heuristics for coreference over non-ACE types. Most of our heuristics are domain independent (e.g. linking the parts of an appositive). Our decision to annotate names and descriptions separately was driven par-

[5] The extraction system detects relations and events in the ACE ontology, but these were not used in the current work.

tially by the need to select the best reference (i.e. name) for co-referent clusters. Adding coreference heuristics for the two new entity types was the single most time-consuming activity, taking 20 of the total 43 hours.

4.3 Relation Extraction

For relation extraction, we used both pattern learning and handwritten patterns. We initialized our bootstrap relation learner with the example instances provided with the domain ontology; Table 3 includes examples of the instances provided to the system as training. Our bootstrap relation learner finds instances of the relation argument pairs in text and then proposes both predicate-argument structure and word-based connections between the arguments as possible new patterns for the relation. The learner automatically prunes potential patterns using information about the number of known-to-be true and novel instances matched by a proposed pattern. By running the pattern extractor over a large corpus, the proposed patterns generate new seeds which are in turn are used to propose new patterns. For this experiment, we incorporated a small amount of supervision during the bootstrapping process (roughly 1 hour total per relation); we also performed ~30 minutes total in pruning domain patterns at the end of learning.

Relation	Arg-1	Arg-2
possTreatmnt	AZT	AIDS
studyDisease	Dr Henri Joyeux	cancer
studyDisease	Samir Khleif	cancer

Table 3: Sample Instances for Initializing Learner

We also used a small amount of human effort creating rules for detecting the relations. The pattern writer was given the guidelines, the examples, and a 2K document background corpus and spent 1 hour per relation writing rules.

The learned patterns use a subset of the full pattern language used by the pattern-writer. The language operates over surface-strings as well as predicate-argument structure. Figure 1 illustrates learned and handwritten patterns for *the possibleTreatmentRelation()*. The patterns in rectangles match surface-string patterns; the tree-like patterns match normalized predicate argument structure. The –WORD- token indicates a wild card of 1-3 words. The blue rectangles at the root of the trees in the handwritten patterns are sets of predicates that can be matched by the pattern.

5 Evaluation

Our question answering evaluation was inspired by the evaluation in DARPA's machine reading program, which requires systems to map the information in text into a formal ontology and answer questions based on that ontology. Unlike ACE, this allows evaluators to measure performance without exhaustively annotating documents, allows for balance between rare and common relations, and implicitly measures coreference without requiring explicit annotation of answer keys for coreference. However because the evaluation only measures performance on the set of queries, many relation instances will be unscored. Furthermore, the system is not rewarded for finding the same relation multiple times; finding 100 instances of *isPossibleTreatment(Penicillin, Strep Throat)* is the same as finding 1 (or 10) instances.

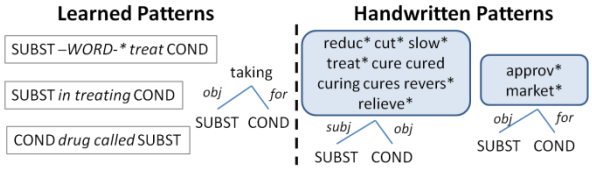


Figure 1: Sample Patterns for *possibleTreatment()*

The evaluation included only queries of the type *Find all instances for which the relation P(X, Z) is true* where one of X or Z is constant. For example, *Find possible treatments for diabetes*; or *What is expected date to market for Abilify?* There were 60 queries in the evaluation set to be answered from a 10K document corpus. To produce a preliminary answer key, annotators were given the queries and corpus indexed by Google Desktop. Annotators were given 1 hour to find potential answers to each query. If no answers were found after 1 hour, the annotators were given a second hour to look for answers. For two queries, both of the form *Find treatments with an expected date to market of MM-YYYY,* even after two hours of searching the annotators were unable to find any answers.[6]

Annotator answers served as the initial gold-standard. Given this initial answer key, annotators reviewed system answers and aligned them with gold-standard answers. System output not aligned with the initial gold standard was assessed as correct or incorrect. We assume that the final gold-standard constitutes a complete answer key, and

[6] Evaluators wanted some queries with no answers.

are thus able to calculate recall for our system and for humans[7]. Because we had only one annotator for each query and because we assumed that any answer found by an annotator was correct, we could not estimate human precision on this task.

Answers can be specific named concepts (e.g. *Penicillin*) or generic descriptions (e.g. *drug, illness*). Given the sentence, *ACME produces a wide range of drugs including treatments for malaria and athletes foot,*' our reading system would extract the relations *responsibleForTreatment(drugs, ACME), possibleTreatment(drugs, malaria), possibleTreatment(drugs, athletes foot).* When a name was available in the document, annotators marked the answer as correct, but underspecified. We calculated precision and recall treating underspecified answers as incorrect and separately calculated precision and recall counting underspecified answers as correct. When treated as correct, there was less than a 0.05 absolute increase in both precision and recall. Unless otherwise specified, all scores reported here use the stricter condition which treats underspecified answers as incorrect.

We also evaluated extracting all information in a small document collection (here human search of the 10k documents does not play a role in finding answers). Individuals were asked to annotate every instance of the 5 relations in a set of 102 documents. Recall, Precision, and F were calculated by aligning system responses to the answer key. System answers that aligned are correct; those that did not are incorrect; and answers in the key that were not found by the system are misses. Unlike the question answering evaluation, this evaluation measures the ability to find every instance of a fact. If the gold standard includes 100 instances of *isPossibleTreatment(Penicillin, Strep Throat),* recall will decrease for each instance missed. The "extraction" evaluation does not penalize systems for missing coreference.

6 Results

6.1 Class Detection

The recall, precision, and F1 for class detection using 10-fold cross validation of the ~1K annotated sentences appear in the 3-5th columns of Table 4. Given the amount of training, our results are lower than in Miller et al (2004) (an F1 of 90 with less than 25K words of training). Several factors could explain this: Finding boundaries and types for descriptions is more complex than for names in English.[8] Our classes, pharmaceutical substances and physiological conditions, may have been more difficult to learn. Our classes are less common in news reporting; as such, both word-class clusters and active learning may have been less effective. Finally, our evaluation was done on a 10-fold split of the active-learning selected data; bias in selecting the data could explain at least a part of our lower performance.

Type	# in GS	Without Lists			With Lists		
		R	P	F	R	P	F
Subst-D	789	77	85	80.8	78	85	81.3
Subst-N	410	70	82	75.5	77	81	78.9
Cond-D	427	72	78	74.9	72	77	74.4
Cond-N	963	80	87	83.4	84	83	83.5

Table 4: Cross Validation: Condition & Substance

We noticed that the system frequently reported country names to be substance-names. Surprisingly, we found that our well-trained name finder made the opposite mistake, occasionally reporting drugs as geo-political entities.

We incorporated lists of known substances and conditions to improve recall. Performance on the same cross-validation split is shown in the final three columns of Table 4. Incorporating the lists led to recall gains for both substance-name and condition-name. Because a false-alarm in class recognition only leads to an incorrect relation extraction if it appears in a context indicating a domain relation, false alarms of classes may be less important in the question answering and extraction evaluations.

6.2 Question Answering and Extraction

Figure 2 and Table 6 show system performance using only handwritten rules (HW), only learned patterns (L), and combining both (C). Figure 2 includes scores calculated with all of the systems' answers (in the dotted boxes), and with just those answers that were deemed useful (discussed be-

[7] The answer key may contain some answers that were found neither by the annotator nor by the systems described here, since the answer key includes answers pooled from other systems not reported in this paper. The system reported here was the highest performing of all those participating in the experiment. Furthermore, if a system answer is marked as correct, but underspecified, the specific answer is put in the key.

[8] English names are capitalized; person names have a typical form and are frequently signaled by titles; organization names frequently have clear signal words, such as Corp.

low). We include annotator recall. Handwritten patterns outperform learned patterns consistently with much higher recall. Encouragingly, however,

1. The combined system's recall and F-Score are noticeably higher for 3 of the relations.
2. The learned patterns generate answers not found by handwritten patterns.
3. The learned patterns have high precision.[9]

There is variation across the different relations. The two best performing relations *possibleTreatment*() and *studiesDisease*() have F1 more than twice as high as the two worst performing relations, *expectedDateToMarket()* and *hasSideEffect()*. This is primarily due to differences in recall.

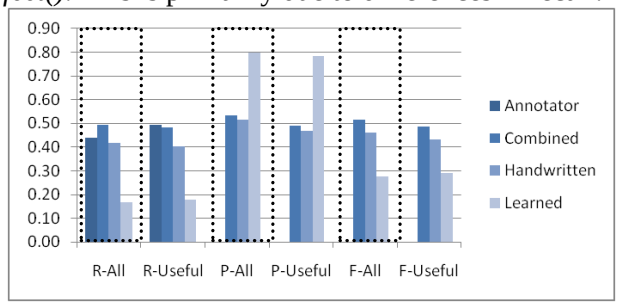


Figure 2: Overall Q/A Performance: All answers in dotted boxes; 'Useful Answers' unboxed

The combined system's recall (0.49), while low, is higher than that of the annotators (0.44). While hardly surprising that a machine can process information much more quickly than a person, it is encouraging that higher recall is achieved even with only one week's effort. In the context of our pooled answer-key, the relatively low recall of both the system and the annotator suggests that there was little overlap between the answers found by the annotator and those found by the system.

As already described, the system answers can include both specific references (e.g. *Prozac*) and more generic references (*the drug*). When a more specific answer is present in the document, generic references have been treated as incorrect. However, sometimes there is not a more specific reference; for example an article written before a drug has been released may never name the drug. Scores reported thus far treat such answers as correct. These answers would be useful when answering more complex queries. For example, given the sentence '*ACME spent 5 years developing a pill to treat the flu which it will release next week,'* extracting relations involving '*the pill'* would allow a system to answer questions that use multiple relations in the ontology to for example ask about *organizations developing treatments for the flu*, or *the expected date of release for ACME's drugs*. However, in our simple question answering framework such generic answers never convey novel information and thus were probably ignored by human annotators.

To measure the impact of treating these generic references as correct,[10] we did additional annotation on the correct answers, marking answers as 'useful' (specific) and 'not-useful' (generic). The unboxed bars in Figure 2 show performance when 'not-useful' answers are removed from the answer-key and the responses. For the four relations where there was a change Table 5 provides the relative change performance when only 'useful' answers are considered. The annotator's recall increases noticeably while the combined system's drops. This results in the overall recall of annotators surpassing that of the combined system.

Relation	Recall				Precision		
	A	C	H	L	C	H	L
possTreat	12	10	10	14	-10	-11	-3
respTreat	9	0	-5	8	-4	-4	-1
studyDis	12	-6	-9	13	-11	-13	0
hasSidEff	3	4	4	4	0	0	0
Total	11	-2	-4	6	-9	-10	-2

Table 5: Relative Change in Recall and Precision When Non-Useful Answers are Removed

Table 7 shows the total number of answers produced by annotators and by each system, as well as the percentage of queries with at least one correct answer for each system. For one relation *expectedDateOnMarket()*, the learned system did not find any answers. This relation had far fewer answers found by annotators and occurred far more rarely in the fully annotated extraction set (see Table 8). Anecdotally, extracting this relation frequently required co-referencing 'it' (e.g. "*It will be released in March 2011"*). Our heuristics for coreference of the new classes did not account for pronouns. Learning from such examples would require coreference during bootstrapping. Most likely, the learned system was unable to generate enough novel instances to continue bootstrapping

[9] The learned patterns' high precision is to be expected for two reasons. First, a few bad patterns were manually removed for each relation. More importantly, the learning algorithm strongly favors high precision patterns because it needs to maintain a seed set with low noise in order to learn effectively.

[10] Generic answers were treated as correct only if a more specific reference was not available in the document.

and was thus unable to learn the relation.

Relation Type (# Queries; # Correct Ans.)	Recall				Precision			F		
	A	C	HW	L	C	HW	L	C	HW	L
possTreatment (10;247)	0.27	0.63	0.50	0.34	0.51	0.47	0.83	0.56	0.48	0.48
respForTreat (15;134)	0.73	0.33	0.24	0.22	0.66	0.78	0.73	0.44	0.37	0.33
expectDateMarkt (11;60)	0.90	0.17	0.17	0.00	0.77	0.83	0.00	0.27	0.28	0
studiesDisease (13;292)	0.23	0.67	0.59	0.09	0.51	0.50	0.79	0.58	0.54	0.16
hasSideEffect (11;104)	0.80	0.10	0.13	0.02	0.83	0.70	1.00	0.17	0.23	0.04
Total (60;837)	0.44	0.49	0.42	0.17	0.53	0.52	0.80	0.51	0.46	0.28

Table 6: Question Answering Results by Relation Type

Relation Type	Total Number of Answers				% Queries with At Least 1 Corr. Ans			
	A	C	HW	L	A	C	HW	L
possTreatment	66	303	261	100	100.0%	90.0%	90.0%	90.0%
respForTreat	98	67	41	40	100.0%	66.7%	60.0%	60.0%
expectDateMarkt	54	13	12	0	72.7%	45.5%	45.5%	0.0%
studiesDisease	68	379	347	33	100.0%	61.5%	46.2%	46.2%
hasSideEffect	83	12	20	2	72.7%	36.4%	45.5%	18.2%
Total	369	774	681	175	90.0%	60.0%	56.7%	43.3%

Table 7: Number of Answers and Number of Queries Answered

Overall, the system did better on relations having more correct answers. Bootstrap learning has an easier time discovering new instances and new patterns when there are more examples to work with. Even a human pattern writer will have more examples to generalize from for common relations.

While *possibleTreatment()* and *hasSideEffect()* have similar F-scores, their performance is very different at the query level. The system was able to find at least one correct answer to every *possibleTreatment*() query; however only 72.7% of the *studiesDisease*() queries were answered.

Table 8 presents results from the extraction evaluation where a set of ~100 documents were annotated for all mentions of the 5 relations. Because every mention in the document set must be found, the system cannot rely on finding the easiest answers for common relations. The results in Table 8 are significantly lower than for the question answering tasks; yet some of the same trends are present. Handwritten rules outperform learned patterns. For at least some relations, the combination of the two improves performance. The three relations for which the learned system has the lowest performance on the question-answering task have the fewest instances annotated in the document set. Fewer instance in the large corpus make bootstrapping more difficult—the learner is less able to generate novel instances to expand its pattern set.

7 Discussion

7.1 Sources of Error

The most common source of error is pattern coverage. In the following figure, the system identified responsibleForTreatment(Janssen Pharmaceutical, Sporanox), but missed the corresponding relation between Novartis and Lamisil.

> *Sporanox is made by Janssen Pharmaceutica Inc., of Titusville, N.J. Lamisil is a product of Novartis Pharmaceuticals of East Hanover, N.J.*

Relation Type	# Relations Found				Recall			Precision			F		
	GS	C	HW	L	C	HW	L	C	HW	L	C	HW	L
possibleTreatment	518	225	187	68	**0.15**	0.10	0.09	**0.34**	0.28	0.66	**0.21**	0.15	0.15
respForTreatment	387	101	77	36	**0.10**	0.08	0.05	**0.41**	0.40	0.50	**0.17**	0.13	0.08
expDateOnMarket	66	13	13	0	***0.06***	***0.06***	0.00	***0.31***	**0.31**	0.00	***0.10***	***0.10***	0.00
studiesDisease	136	95	91	4	0.08	***0.09***	0.00	0.12	**0.13**	0.00	0.10	**0.11**	0.00
hasSideEffect	256	26	25	2	***0.04***	***0.04***	0.00	0.39	0.40	0.50	***0.07***	***0.07***	0.01

Table 8: Extraction Results on the 102 Document Test Set Annotated for All Instances of the Relations

Missed class instances contribute to errors, sometimes originating in errors in tokenization (e.g. not removing the '_' in each drug name in a bulleted list of the form "*_Trovan, an antibiotic...; etc.)* However, many drug-names are simply missed:

> *Pfizer also hopes to introduce Pregabalin next year for treatment of neuropathic pain, epilepsy and anxiety…Other deals include co-promoting Rebif for multiple sclerosis with its discoverer, Serono, and marketing Aricept for Alzheimer's disease with its developer, Eisai Co.*

The system correctly identifies *Rebif* and *Aricept* as drugs, but misses *Pregabalin* and *Serono*. In both misses, the immediately preceding and following words provide little evidence that the word refers to a drug rather than some other product. Substance detection might be better served with a web-scale, list-learning approach like the doubly anchored patterns described in (Kozareva et al., 2008). Alternatively, our approach may need to be extended to include a larger context window.

7.2 Learned Patterns

One of the ways in which learned patterns supplement handwritten ones is learning highly specific surface-string patterns that are insensitive to errors in parsing. Figure 3 illustrates two examples of what appear to be easy cases of *possibleTreatment*(). Because the handwritten patterns are not exhaustive and make extensive use of syntactic structure, parse errors prevented the system based on handwritten rules from firing. Learned surface-string patterns were able to find these relations.

Even when the syntactic structure is correct, learned patterns capture expressions not common enough to have been noticed by the rule writer. For example, while the handwritten patterns included '*withdrew'* as a predicate indicating a company was responsible for a drug, they did not include '*pulled.'* By including '*pulled'*, learned patterns extracted *responsibleForTreatment()* from '*American Home Products pulled Duract, a painkiller.'* Similarly, the learned patterns include an explicit pattern '*CONDITION drug called SUBSTANCE'*, and thus extracted a *possibleTreatment()* relation from '*newly approved narcolepsy drug called modafinil'* without relying on the coreference component to link *drug* to *modafinil*.

Handwritten Patterns

Despite the examples above of successfully learned patterns, handwritten patterns perform significantly better. In the active-learning context used for these experiments, the handwritten rules also required *less* manual effort. This comparison is not entirely fair-- while learned patterns required more hours, supervising the bootstrapping algorithm requires no training. The handwritten patterns, in contrast, require a trained expert.

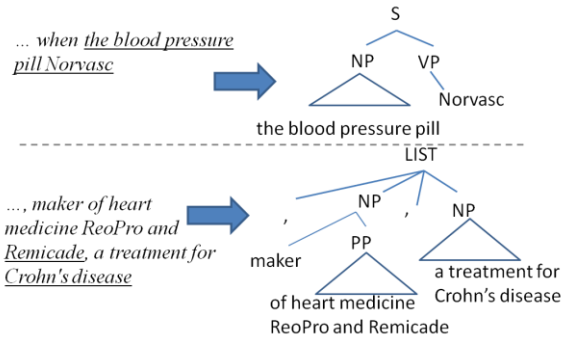


Figure 3: Extractions Missed by Handwritten Rules & the Erroneous Parses that Hid them

While handwritten rules and learned patterns use the same language, they make use of it differently. The handwritten patterns group similar concepts together. A human pattern writer adds relevant synonyms, as well as words that are not synonymous but in the pattern context can be used interchangeably. In Figure 4, the handwritten patterns include three word-sets: (*patient**, *people*, *participant**); (*given*, *taken*, *took*, *using*); and (*report**, *experience**, *develop**, *suffer**). The '*' serves as a wild-card to further generalize a pattern. The word-sets in Figure 4 illustrate challenges for a learned system: the words are not synonyms, but rather are words that can be used to imply the relation.

A human pattern writer frequently generates new classes not in the domain ontology. In Figure 4, the circled patterns form a class of 'people taking a substance.' The handwritten patterns for *studiesDisease*() include classes targeting scientists and researchers. These classes are not necessarily triggered by nouns. Such classes allow the pattern writer to include complex patterns as in Figure 4 and to write relatively precise, but open-ended patterns such as: *if there is a single named-drug and a named, non-side-effect disease in the same sentence, the drug is a treatment for the disease.*

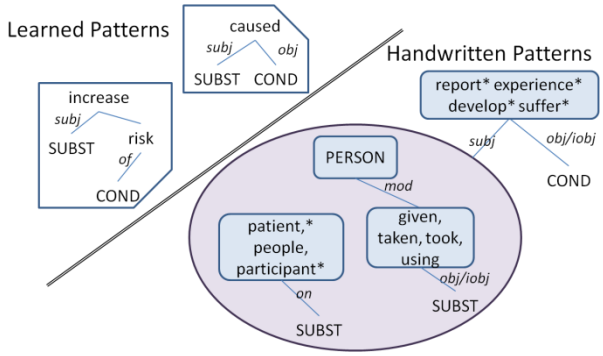


Figure 4: Learned and Handwritten Patterns for *hasSideEffect*()

A final difference between handwritten and learned patterns is the level of predicate-argument complexity used. In general, handwritten patterns account for larger spans of predicate argument structure while learned patterns tend to limit themselves to the connections between the arguments of the relation with minor extensions.

8 Conclusions and Lessons Learned

First, it is encouraging that the synthesis of learning algorithms and handwritten algorithms can achieve an F1 of 0.51 in a new domain in a week (<50 hours of effort). *Second*, it is exciting that so little training data is required: ~20 relation pairs out of context (~2.5 hours of effort) and ~6 hours of active learning for the new classes.

Third, the effectiveness of learning algorithms is still not competitive with handwritten patterns based on predicate-argument structure (~5 hours of effort on top of active learning for entities). Though the learned patterns have high precision (0.80 on average), recall is low (0.17) and varied greatly across the relations. Though the dominant factor in missing relations is pattern coverage, missing instances of classes contributed to low recall. Comparing learned patterns to manually written patterns, (1) synonyms or other lexical alternatives that a human pattern writer would include, (2) the creation of subclasses for argument types, and (3) the scope of patterns[11] are each major sources of the disparity in coverage. Research on learning approaches to raise recall without significant sacrifice in precision seems essential.

Fourth, despite the disparity in performance of learned versus manual patterns, and despite the low recall of learned patterns, the combined system's recall and F-Score are higher for three of the relations because the learned patterns generated answers not found by handwritten patterns. We found examples where highly specific, learned, surface-level patterns (lexical patterns) occasionally found information missed by handwritten patterns due to parsing errors or general low coverage.

Fifth, the effort for coreference was the most time-consuming, given that every new relation contained at least one of the new argument types. While we included this in our estimate of domain adaptation, the infrastructure we built is domain generic. Improving generic coreference will reduce domain specific effort in future.

Perhaps *most significant* of all, running a complete experiment from definition of the domain through creation of training data and measurement of end-to-end performance of the system can be completed in a month. The ability to rapidly, cheaply, and empirically measure the impact of extraction research could prove a significant spur to research across the board.

These experiments suggest three possible directions for improving the ability to quickly develop information extraction technology for a new set of relations: (1) reducing the amount of supervision provided to the bootstrap-learner; (2) improving the bootstrapping approach to reach the level of recall achieved by the human pattern writer eliminating the need for a trained expert during domain adaptation; and (3) focusing improvements to the bootstrapping approach on techniques that allow it to find more of the instances missed by the pattern writer, thus improving the accuracy of the hybrid system.

[11] Learned patterns tend to focus on the structure that appears between the two arguments, rather than structure surrounding the left and right arguments.

Acknowledgments

This work was supported, in part, by DARPA under AFRL Contract FA8750-09-C-179. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited) Thank you to the reviewers for your insightful comments and to Michelle Franchini for coordinating the assessment effort.

References

E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *Proceedings of the ACM Conference on Digital Libraries*, pp. 85-94, 2000.

A. Blum and T. Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, July 1998.

E. Boschee, V. Punyakanok, R. Weischedel. An Exploratory Study Towards 'Machines that Learn to Read'. *Proceedings of AAAI BICA Fall Symposium*, November 2008.

J. Chen, D. Ji, C. Tan and Z. Niu. (2006). Relation extraction using label propagation based semi-supervised learning. *COLING-ACL 2006*: 129-136. July 2006.

M. Freedman, E. Loper, E. Boschee, and R. Weischedel. Empirical Studies in Learning to Read. Proceedings of NAACL 2010 Workshop on Formalisms and Methodology for Learning by Reading, pp. 61-69, June 2010.

W. Li and A. McCallum. Rapid development of Hindi named entity recognition using conditional random fields and feature induction. Transactions on Asian Language Information Processing (TALIP), Volume 2 Issue 3 September, 2003.

R Grishman and B. Sundheim. Message Understanding Conference-6 : A Brief History", in COLING-96, *Proc . of the Int'l Conj. on Computational Linguistics*, 1996.

Z. Kozareva and E. Hovy. Not All Seeds Are Equal: Measuring the Quality of Text Mining Seeds. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, June, 2010, pp. 618-626.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056.

J. May, A. Brunstein, P. Natarajan, and R. Weischedel. Surprise! What's in a Cebuano or Hindi Name? Transactions on Asian Language Information Processing (TALIP), Volume 2 Issue 3 September, 2003.

D. Maynard, V. Tablan, K. Bontcheva, and H. Cunningham. Rapid customization of an information extraction system for a surprise language. Transactions on Asian Language Information Processing (TALIP), Volume 2 Issue 3 September, 2003.

S. Miller, J. Guinness, and A. Zamanian, "Name Tagging with Word Cluster and Discriminative Training", *Proceedings of HLT/NAACL* 2004, pp. 337-342, 2004

T. Mitchell, J. Betteridge, A. Carlson, E. Hruschka, and R. Wang. "Populating the Semantic Web by Macro-Reading Internet Text. Invited paper, *Proceedings of the 8th International Semantic Web Conference (ISWC 2009).*

NIST, ACE 2007: http://www.itl.nist.gov/iad/mig/tests/ace/2007/software.html

P. Pantel and M. Pennacchiotti. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of Conference on Computational Linguistics / Association for Computational Linguistics (COLING/ACL-06)*. pp. 113-120. Sydney, Australia, 2006.

L. Ramshaw , E. Boschee, S. Bratus, S. Miller, R. Stone, R. Weischedel, A. Zamanian, "Experiments in multi-modal automatic content extraction", Proceedings of Human Technology Conference, March 2001.

D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 41–47, Philadelphia, PA, 2002.

E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044-1049, 1996.

S. Sekine and R. Grishman. Hindi-English cross-lingual question-answering system. Transactions on Asian Language Information Processing (TALIP), Volume 2 Issue 3 September, 2003.

G. Zhou, J. Li, L. Qian, Q. Zhu. Semi-Supervised Learning for Relation Extraction. *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*. 2008.

Discovering Relations between Noun Categories

Thahir P Mohamed *
University Of Pittsburgh
pmthahir@gmail.com

Estevam R Hruschka Jr.
Federal University of Sao Carlos
estevam@cs.cmu.edu

Tom M Mitchell
Carnegie Mellon University
tom.mitchell@cs.cmu.edu

Abstract

Traditional approaches to Relation Extraction from text require manually defining the relations to be extracted. We propose here an approach to automatically discovering relevant relations, given a large text corpus plus an initial ontology defining hundreds of noun categories (e.g., Athlete, Musician, Instrument). Our approach discovers frequently stated relations between pairs of these categories, using a two step process. For each pair of categories (e.g., Musician and Instrument) it first co-clusters the text contexts that connect known instances of the two categories, generating a candidate relation for each resulting cluster. It then applies a trained classifier to determine which of these candidate relations is semantically valid. Our experiments apply this to a text corpus containing approximately 200 million web pages and an ontology containing 122 categories from the NELL system [Carlson et al., 2010b], producing a set of 781 proposed candidate relations, approximately half of which are semantically valid. We conclude this is a useful approach to semi-automatic extension of the ontology for large-scale information extraction systems such as NELL.

1 Introduction

The Never-Ending Language Learner (NELL) (Carlson et al., 2010b)) is a computer system that learns continuously to extract facts from the web. NELL is given as input an initial ontology that specifies the semantic categories (e.g. city, company, sportsTeam) and semantic relations (e.g. hasOfficesIn(company,city), teamPlaysInCity(sportsTeam,city)) it must extract from the web. In addition, it is provided 10-20 seed positive training examples for each of these categories and relations, along with hundreds of millions of unlabeled web page. Given this input, NELL applies a large-scale multitask, semisupervised learning method to learn to extract new instances of these categories (e.g., city("London")) and relations (e.g., teamPlaysInCity("Steelers","Pittsburgh")) from the web. During the past 17 months NELL has been running nearly continuously, learning to extract over 600 categories and relations, and populating a knowledge base containing over 700,000 instances of these categories and relations with a precision of approximately 0.85[1].

This paper considers the problem of automatically discovering new relations to extend the ontology of systems such as NELL, enabling them to increase over time their learning and extraction capabilities. More precisely, we consider the following problem:

Input:
- An ontology specifying a set of categories
- A knowledge base containing instances of these categories (perhaps including errors)
- A large text corpus

Output:
- A set of two-argument relations that are frequently mentioned in the text corpus, and whose argument types correspond to categories in the input ontology (e.g., RiverFlowsThroughCity(<River>,<City>).
- For each proposed relation, a set of instances (i.e. RiverFlowsThroughCity("Nile","Cairo")).
- For each proposed relation, a set of text extraction patterns that can be used to extract additional instances of the relation (e.g., the text "X in the heart of Y", where X is a known river, and Y a known City, suggests extracting RiverFlowsThroughCity(X,Y)).

Note the above inputs are easily available from NELL in the form of its existing ontology and extracted knowledge base. Note also that the outputs

* Thahir P. Mohamed is currently at Amazon Inc.

1 NELL's extracted knowledge can be viewed and downloaded at http://rtw.ml.cmu.edu.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1447–1455,
Edinburgh, Scotland, UK, July 27–31, 2011.

of our system are sufficient to initiate NELL's learning of additional extraction methods to further populate each proposed relation. One goal of this research is to create a system that can provide NELL with an ongoing set of new learning and extraction tasks. The system is called OntExt (Ontology Extension System)

Table 1 shows a sample of successful relations and corresponding relation contexts and sample seed instances generated by OntExt.

Table 1. Examples of valid relations (generated by OntExt), their text extraction patterns and extracted instances.

name(category1-main context-category2)	Extraction patterns	Seed Instances
River -in heart of- City	'in heart of' 'in the center of' 'which flows through'	"Seine, Paris" "Nile, Cairo" "Tiber river, Rome" "River arno, Florence"
Food -to produce- Chemical	'to produce' 'to make' 'to form'	"Salt, Chlorine" "Sugar, Carbon dioxide" "Protein, Serotonin"
StadiumOrVenue -in downtown- City	'in downtown'	"Ford field, Detroit" "Superdome, New Orleans" "Turner field, Atlanta"
Disease -caused by- Bacteria	'caused by' 'is the causative agent of' 'is the cause of'	"pneumonia, legionella" "mastitis, staphylococcus aureus" "gonorrhea, neisseria gonorrhoeae"
Disease -destroys- CellType	'destroys' 'attacks'	"alzheimer, brain cells" "vitiligo", melanocytes" "aids, lymphocytes"
County -county- StateOrProvince	'county' 'county of' 'county in'	"sufolk, massachusetts" "marin, california" "sussex, delaware" "osceola, michigan"

2 Background

Traditional Relation Extraction

We define Traditional RE systems as those that require the user to specify information about the relations to be learned. For instance, SnowBall (Agichtein and Gravano 2000) & CPL (Carlson et al. 2009) are bootstrapped learning systems that require manual input of relation predicates. In these systems, for each relation predicate, the relation name (e.g. City 'Capital of' Country), the seed instances and the category type (e.g. City, Country, Celebrity etc) are provided (for domain and range). In CPL (Carlson et al. 2009), learning of relation/category instances is coupled by using constraints such as mutual exclusion relationships among the predicates. The authors show that this coupling reduces semantic drift, which commonly occurs with bootstrapping systems, thus leading to improved precision. CPL achieved 89% precision for the relation instances extracted (Carlson, Betteridge et al. 2009). KNOWITALL (Etzioni, Cafarella et al. 2005) is a web-scale relation extraction system, which requires as input the relation names. Hence, in these "traditional relation extraction" methods, the need to manually define the relations to be extracted makes it difficult to work in applications having thousands of possible relation predicates.

2.1 Open Relation Extraction

Open RE methods do not require a user to manually specify the information about the relations to be learned, such as their names, seed examples, etc. TextRunner (Banko, Cararella et al. 2007) is such an Open Information Extraction system that retrieves from the web millions of relational tuples between noun phrase entities. TextRunner uses a deep linguistic parser to perform self-supervised learning and extracts a positive set (i.e. valid relation between entities) and a negative set (i.e. invalid relationships) of relational tuples based on certain heuristics. Then, a Naive Bayes classifier is built having features such as part-of-speech tags of the words in the relation tuples, number of tokens, stopwords etc., and uses the labeled instances as the training set. This classifier runs on sentences from a web corpus to extract millions of relational tuples. However, of the 11 million high confident relational tuples extracted by this system only 1 million were concrete facts (Banko, Cararella et al. 2007). Of these concrete facts 88% were estimated to be correct. For instance, (Mountain View, headquarters of, Google) is a tuple representing a valid concrete fact. The remaining 90% of the tuples are abstract or do not have well-formed arguments or well-formed relations. For instance, (Einstein, derived, theory) is an abstract tuple as it does not

have enough information to indicate a concrete fact (Banko, Cararella et al. 2007) because the specific theory which Einstein derived is missing in that tuple. In the tuple (45, 'went to', 'Boston'), one of the arguments (i.e. 45) is not well formed.

In (Banko and Etzioni, 2008) a Conditional Random Field (CRF) classifier is used to perform Open Relation Extraction which improves by more than 60% the F-score achieved by the Naive Bayes model in the TextRunner system. However the CRF approach does not solve the problem associated with extraction of abstract/non-well formed tuples. Further, in the same work, it is shown that Open RE has a much lower recall in comparison to Traditional RE systems. On four common relations (Acquisition, Birthplace, InvetorOf, WonAward), Open RE attained a recall of 18.4% in comparison to 58.4% achieved by Traditional RE (Banko and Etzioni 2008). Both Open RE systems discussed (Banko, Cararella et al. 2007; Banko and Etzioni 2008) do not perform learning of the category type of the entities involved in the relations. They are single-pass and do not perform continuous learning to improve/extend on what has been learnt.

2.2 Unsupervised Methods to Extract Relations between Named Entities

In general, traditional RE methods extract concrete facts and have much higher recall for a given relation, than Open RE methods. This is due to the knowledge fed into Traditional RE methods such as the category type of the entities in the relation and seed instances for the relation. Traditional RE methods require the relations to be manually defined and extract instances only for them. Open RE methods, on the other hand, do not require any such domain specific knowledge to be manually input. They extract instances for a wide spectrum of relations that are not manually pre-defined.

To overcome the drawbacks of using Traditional and Open RE methods, some researchers have used unsupervised learning methods to automatically generate new relations (with seeds and contexts) between specific categories. These automatically generated relations can then be used as input to Traditional RE systems.

Hasegawa et.al (Hasegawa, Sekine et al. 2004), propose an unsupervised clustering based approach. One feature vector for each co-occurring NE pair is formed based on the context words in which the NE pair co-occurs. Then, a cosine-similarity metric is applied to each pair of feature vectors to generate a "NE-pair x NE-pair" matrix. Clustering is done on this matrix and each cluster of NE-pairs corresponds to a relation predicate.

The work by Zhang et.al (Zhang, Su et al. 2005) generates a shallow parse tree for each sentence containing a NE pair to generate relation instances. A tree similarity metric is used to cluster the relation instances. This method gives improved F-score over Hasegawa et.al (Hasegawa, Sekine et al. 2004). Further they use a specialized NE tagger built to recognize entities that belong to specific predefined categories. The aforementioned methods (Hasegawa, Sekine et al. 2004) (Zhang, Su et al. 2005) were tested on a news corpus to identify relations between only a couple of pairs of entity types (Person-GeoPoliticalEntity and Company-Company).

Both of these methods cluster NE-pairs primarily based on lexical similarity of the context words connecting the entities. Hence NE-pairs connected by lexically different but semantically similar context patterns (e.g. river 'in heart of' city and river 'flows through' city) would probably not get clustered together. The web data is, however, much noisier and has a larger number of entity types (i.e. category predicates), thus, another issue is that for web scale data *NE pairs X NE pairs* similarity matrix would not be scalable for many thousands of NE-pairs.

3 Ontology Extension System - OntExt

The OntExt system for ontology extension, proposed in this paper, combines characteristics from both "Traditional RE" and "Open RE," to discover new relations among categories that are already present in the ontology, and for which many instances have already been extracted.

Our proposed method for automatic relation extraction offers the following advantages over the methods discussed above.

- The key idea in our approach is to make use of redundancy of information in web data - the same relational fact is often stated multiple times in large text corpora, using different context patterns. We use this redundancy to cluster together context patterns which are semantically similar although they may be lexically dissimilar.

- Instead of clustering on the *'NE-pairs X NE-pairs'* matrix, clustering is done on a *'Context-pattern X Context-pattern'* matrix. This is much more scalable as the context patterns are fewer in number and since our method applies several criteria to prune out irrelevant patterns.
- To accommodate errors in the input category instances and ambiguity in web data, we build a classifier which learns to distinguish valid relations from semantically invalid relations.

OntExt has 3 components. 1) It starts exploring a large web corpus and 2) category instances extracted by CPL to generate new relations. After the relations are generated, 3) a classifier is developed to classify semantically valid relations.

3.1 Pre-processing

Following along the same strategy used in [Carlson et al., 2010], OntExt uses as input a corpus of 2 billion sentences, which was generated by using the OpenNLP[2] package to extract, tokenize, and POS-tag sentences from the 500 million web page English portion of the ClueWeb09 data [Callan and Hoy, 2009]. Before performing relation extraction, this corpus is preprocessed. First, sentences which contain a pair of known category instances are retrieved (e.g. the sentence "Ottawa is the capital of Canada.", where 'Ottawa' is a known instance of the 'City' category and 'Canada' is a known instance of 'Country'). For every category pair (e.g. <City, Country>) the sentences containing known instances of both categories are grouped into a set S. The text between the two instances is called the 'context pattern' (e.g. 'is the capital of' is a context pattern). Three types of pruning are done on this set S.

1. If the context pattern is a rare one (i.e. if the context pattern occurs in less than a threshold number of sentences), all sentences with that context pattern are removed. Thus we retain only frequently occurring contexts. We use a threshold requiring at least 5 sentences in the experiments presented in Section 4.
2. Context patterns which co-occur with very few instances of either category type are removed. For example, the category pair <Vehicle,SportsTeam> has several sentences such as 'Car was engulfed in flames', 'Truck was engulfed in flames' etc. Note that Flames (Calgary Flames) is a SportsTeam. But here flames clearly does not refer to a Sportsteam. This context 'was engulfed in' connects several instance of a 'Vehicle' category to a single instance of SportsTeam instance. Hence all sentences with this context are removed. Note this context would not have been removed in step 1 as that is just a threshold on the number of sentences in which any pair occurs. We use a threshold requiring at least 3 distinct instances of both the domain and the range, for each context.
3. Banko et.al, 2008 show that most binary relational contexts fall under certain types of lexico-synctatic patterns. They include context patterns like 'C1 Verb C2', 'C1 NP Prep C2', 'C1 Verb Prep C2' and 'C1 to Verb C2' (C1 and C2 are category instances). Hence context patterns which do not fall under the above types are removed from the set S as they are not likely to produce relation instances.

3.2 Relation Generation

From the previous pre-processing step OntExt retrieves for each category pair a pruned set S' of sentences. Each sentence has a pair of category instances and the context connecting them.

Algorithm 1: Relation Generator

Input: One pair of Categories (C1, C2) and set of sentences, each containing a pair of instances known to belong to C1 and C2. The phrase connecting the instances in the sentence is the context.
Output: Relations and their seed instances

Steps:

1. From the input sentences, build a Context by Context co-occurrence matrix (Shown in figure 1). The matrix is then normalized.
2. Apply K-means clustering on the matrix to cluster the related contexts together. Each cluster corresponds to a possible new relation between the two input categories. (Weka Machine Learning package [Hall et al., 2009] was used to perform K-means clustering. The value of K was set to 5 based on trial and error experiments.)

2 http://opennlp.sourceforge.net.

3. Rank the known instance pairs (belonging to C1,C2) for each cluster and take the top 50 as seed instances for the relation

The key data structure used by OntExt is a co-occurrence matrix of the contexts for each category pair, as shown in Figure 1. In this matrix, each cell corresponds to the number of pairs of category instances that both contexts co-occur with (e.g. the sentences "Vioxx can cure Arthritis" and "Vioxx is a treatment for Arthritis" provide a case where the 2 contexts 'can cure' and 'is a treatment for' co-occur with an instance pair [Vioxx, Arthritis]). Initially, the value of Matrix(I,j) is the number of category instance pairs that occur with both context i and context j. We then normalize each cell in the matrix, dividing it by by the total count for its row.

$$Matrix(i,j) = \frac{Matrix(i,j)}{\sum_{j=0}^{N} Matrix(i,j)}$$

We also give higher weight to contexts which co-occur with only a few contexts over ones which are generic and co-occur with most contexts.

$$Matrix(i,j) = Matrix(i,j) * \frac{N}{|\{Context(j) : Matrix(i,j) > 0\}|}$$

Where N is the total number of contexts, and |{Context(j) : Matrix(i,j) > 0}| refers to the number of cells in the row Matrix(i) which are greater than zero.

For example, for the <drug, disease> category pair after 122 contexts were obtained after preprocessing. Contexts such as 'to treat', 'for treatment of', 'medication' which all indicate the same relation (drug-to treat-disease) have high co-occurrence values (see Figure 1). Similarly contexts such as 'can cause', 'may cause', 'can lead to' (indicating the relation drug-can cause-disease) have high co-occurrence values (see Figure 1). When OntExt performs clustering on this co-occurrence matrix the contexts with large co-occurrences get clustered together. Each cluster is then used to propose a possible new relation. The centroid of each cluster is used to build the relation name. If the centroid of a cluster is the context 'for treatment of', then the relation name is 'drug-for-treatment-of-disease'.

OntExt next generates seed instances for the proposed relation. The seed instances which co-occur with contexts corresponding to the cluster centroid or close to centroid will be best representative of the relation. So the strength of the seed instance is inversely proportional to the standard deviation of the context from the centroid of the relation contexts cluster. Also the strength of the seed instance is directly proportional to the number of times it co-occurs with the context.

Contexts/ Contexts	may cause	can cause	can lead to	to treat	for treatment of	medication
may cause	0.176	0.074	0.030	0.015	0.011	0.000
can cause	0.051	0.150	0.039	0.018	0.013	0.010
can lead to	0.034	0.064	0.189	0.019	0.021	0.018
to treat	0.006	0.011	0.007	0.109	0.043	0.015
for treatment of	0.005	0.008	0.008	0.045	0.086	0.023
medication	0.000	0.011	0.009	0.030	0.036	0.111

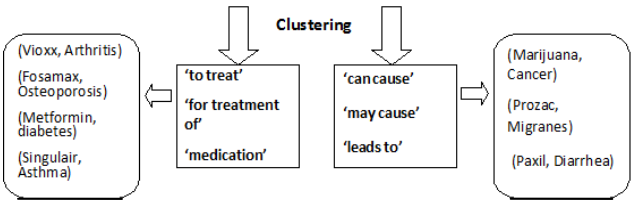


Figure 1: This figure shows the Context by Context sub-matrix (with 6 contexts) for the category pair (Drug, Disease) and the seed instances for each relation. As described in the text, each entry gives the normalized count of the number of known <drug, disease> pairs that occur with both the row context and the column context.

To summarize, each seed instance *s* (pair of category instances) is weighted as follows

$$\sum_{c \in Pattern\ cluster} Occ(c,s)/(1 + sd(c))$$

Where,

Pattern_cluster is the cluster of pattern contexts for this given relation

Occ(c,s) is the number of times instance 's' co-occurs with the pattern context 'c'

sd(c) is the standard deviation of the context from the centroid of the pattern cluster.

Using this metric the instances are ranked and the top 50 are output as initial seed instances for the proposed relation.

3.3 Classifying semantically valid relations

More than half of the relations generated in the previous step are invalid due to the following reasons

1. Error in category instances: The category instances input to OntExt come from NELL. In the version of the knowledge base used in these experiments, the accuracy of these instances was 78%. Due to the erroneous category instances some invalid relations are generated by OntExt. For instance the generated relation, 'condiment-wearing-clothing' with seeds (pig,dress), (rabbit,pants) etc. Here 'pig' and 'rabbit' were incorrectly identified by NELL as instances of 'condiment'.
2. Semantic Ambiguity: Consider the generated relation 'bakedgood-baking-magazine' with instances (cookies,time), (cupcakes, people), etc. Here the instances 'time' and 'people' do not refer to magazines, although they can in general. Due to the semantic ambiguity of these instances this invalid relation got generated
3. Semantically Incomplete relations: Some of the generated relations require a third entity or some more contextual information, in order to be considered semantically valid. For instance, 'personUs-said-company' or 'newspaper-is-reporting-that-company'. These don't stand by themselves as two-argument relational facts and need more information to be complete
4. Illogical relations: Some generated relations simply have no real semantic meaning. These relations are generated due to the category instances appearing together in some unrelated contexts. E.g. the generated relation 'date-starting-date' with seeds such as (Wednesday, June), (friday, July) and the relation 'country-minister-of- economicsector' with seeds (japan,agriculture), (india, industry).

The introduction of these invalid relations can adversely affect the performance of NELL. However, it is a challenging problem to develop automated ways to distinguish between valid and invalid relations without any domain specific knowledge. To approach this problem, we identified a set of features which can help characterizing valid and invalid relations, and which can be generated automatically. Below is a description of the features and the intuition behind their use for this classification task.

Each generated relation has a pair of category types (C1, C2), a corresponding set of seed instances (which are pairs of instances belonging to C1 and C2) and pattern contexts connecting C1 and C2. Let N be the number of seed instance pairs and N1 and N2 be number of unique instances (out of these N instance pairs) belonging to categories C1 and C2 respectively.

1. Normalized frequency count: The frequency count of each category instance is obtained from the corpus and normalized by the category instance with maximum count. For a given relation, a feature is generated by averaging the normalized frequency counts of the instances belonging to C1. Another similar feature is generated for C2 following the same strategy. For example the relation <Profession 'believe that' Movie> was generated due to common words like 'predator', 'earthquake' being identified as movie names out of context. These features can help identify such invalid relations.
2. Distribution of extraction patterns: NELL learns instances as well as extraction patterns for each category (e.g. the category Actor has extraction patterns such as '_ got an Oscar award', '_ is the movie's lead actor'). If a category instance co-occurs in the web corpus with several extraction patterns belonging to other categories, then that instance has large ambiguity. We measure ambiguity of an instance (i) belonging to category 'C' with respect to another category 'M' (where M is not a sub type or super type of 'C') as

   $$\text{Ambiguity}(i,M) = \frac{\#\text{ of extraction patterns in 'M' that 'i' co-occurs with}}{\#\text{ of extraction patterns in 'C' that 'i' co-occurs with}}$$

   We measure the average ambiguity for the set of instances (of size N) belonging to category C in the generated seeds as follows,

   $$Max_M\left(\sum_{i \in C} Ambiguity(i,M)/N\right)$$

   Two features are generated for categories C1 and C2 in the relation.
3. Relationship characteristics: We identified a few characteristics of the relation which help in identifying valid relations. If in the generated relation, most instances of C1 co-occur only

with very few instances of C2 (or vice versa) then the relation could be weak. For example, <Organization 'Provides' EconomicSector> - the instance 'Information' (of category EconomicSector) connects to a large percentage of items in the category 'Organization' but does not express a meaningful relation. So we consider the instance (in this example 'Information', let us call it 'maxconnect_instance') co-occurring with maximum number of instances of the other category. The percentage of instances it co-occurs with from among the total number of instances of the other category which are part of the seed instances is taken as a feature. Also if that instance is a very common word (like 'information' which in several contexts does not refer to 'EconomicSector') then this could indicate the presence of an invalid relation. So the normalized frequency count of this instance (maxconnect_instance) is taken as another feature.

4. Pattern Contexts: The number of pattern contexts attained through pattern clustering for the relation is taken as another feature. The presence of several pattern contexts connecting the instances between the two categories could indicate that the relation is a valid one. The presence of Hearst patterns (Hearst M, 1992) referring to a hyponym ("is-a") relation in pattern contexts indicates the possibility of a valid relation, and is taken as another feature

   Another feature is regarding how specific is the context pattern to this relation. If the same context connects say C1 instances to instances of several other categories apart from C2, then this context is not unique to this relation and might not indicate a meaningful valid relationship. So the ratio of the number of instances in C2 connected to C1 versus the number of instances from all categories connected to C1 by the most significant pattern context (i.e. centroid in pattern cluster) is taken as a feature. A similar feature is generated for C2 as well.

4 Experimental Setup and Results

4.1 CPL System

CPL (Carlson et al., 2010) is a semi-supervised learning system which takes in an input ontology (containing category and relation predicates and corresponding seed instances) and constraints (such as Mutual exclusion rules between predicates). The system iteratively extracts patterns and instances for the category/relation predicates from a web corpus of around 500 million web pages. CPL is one learning component in NELL (the Never Ending Language Learner) (Carlson et al., 2010b).

4.2 Relation Generation:

We use approximately 22,000 category instances belonging to 122 categories extracted by CPL at the end of its 20th iteration and the web corpus as input to perform the co-clustering described in Section 3.2 and generate the new relations. The process generated 781 relations. For each relation, the relation name, types of the categories involved in the relation and the seed instances and patterns for each relation were generated. Table 1 in section 1 shows a sample of valid relations generated by this method.

Tables 2, 3, 4 and 5 show invalid relations for each type of invalidity, "Error in the Category Instances", "Semantic Ambiguity", "Semantically Incomplete Relations" and "Illogical Relations" respectively. More specifically, Table 2 shows a sample of relations generated due to an entity being labeled incorrectly as to belong to a category. The incorrect category instances are in italics.

Table 3 presents a sample of relations which were generated because of semantic ambiguity. Instances with ambiguity are in italics.

Table 4 shows some of the generated relations which are semantically incomplete.

Table 5 presents samples of illogical relations which do not establish any concrete fact.

Table 2. Examples of Incorrect category instances.

name(category1 -main context- category2)	Relation Contexts	Seed Instances
SportsGame -Beating- Country	'beating'	"tournament,Sri Lanka" "champions, France" "match, canada"
Animal -will eat- Condiment	'will eat' 'eating'	"wolf, sheep" "fox, rabbit" "lion, lamb"

Table 3. Examples of Semantically Ambiguous relations.

Name	Relation Contexts	Seed Instances
Bird -play- City	‘play’	"Cardinals, Atlanta" "Ravens, Miami" "Eagles, Chicago"
BakedGood -baking- Magazine	‘baking’	"time, cakes" "people, cookies"

Table 4. Examples of semantically incomplete relations.

Name	Relation Contexts	Seed Instances
Personus acknowledged Date	‘acknowledged’ ‘warned’ ‘met’	"mr obama, tuesday" "george w . bush, tuesday" "al gore, thursday"
NewsPaper -is reporting that- Company	‘is reporting that’ ‘writes that’ ‘reported that’	"financial times, apple" "wall street journal, gm" "wall street journal, yahoo"

Table 5. Examples of relations representing facts that are not concrete.

Name	Relation Contexts	Seed Instances
Emotion -of living in- StateOrProvince	‘of living in’	"joy, california" "excitement, colorado" "fear, iowa"
BodyPart -to keep- BodyPart	‘to keep’ ‘guard’	“hand, eye” “nose, throat” “eye, brain” “elbow, hand”

4.3 Relation Classification:

To determine the feasibility of automatically classifying OntExt’s proposed relations as valid or invalid, we trained and tested a classifier using the features described above, using manually assigned class label for some of the generated relations (252 relations) as valid or invalid (the criteria for which was explained before). 115 of these 252 relations were found to be valid by manual evaluation. This shows the need for a machine learning classifier to identify valid/invalid relations. The various features described earlier (such as normalized frequency count, relationship characteristics, pattern context features, distribution of extraction patterns) were generated for each relation. Ten-fold cross validation experiments were carried out with various classifiers. A Random Forest classifier performed the best. Precision, recall and ROC-area is shown in the table below (ROC area is the area under the ROC curve which plots the classifier performance by having the True Positive Rate on the Y-axis and False Positive Rate on the X-axis).

Table 6. Classifier performance.

RelationType	Precision	Recall	ROC Area
Valid	71.6	72.2	0.804
Invalid	76.5	75.9	0.804
Weighted Avg.	74.2	74.2	0.804

These results indicate that the system is able to learn to identify semantically valid relations without using any manually input information. The valid relations generated can be input to NELL, allowing it to iteratively learn additional instances for each proposed relation.

5 Conclusion and Future work:

Open Relation Extraction and Traditional Relation Extraction have their respective strengths and weaknesses. The OntExt system proposed in this work combines the strengths of both of those methods. The relation predicates automatically generated by our approach are typed, have a meaningful name identifying the relation, and are accompanied by suggested context patterns and seed instances. These relations can be input to NELL to learn more instances for the relation. We propose in the future to integrate this relation generation system into NELL, to iteratively extend NELL’s initial ontology, providing an ongoing stream of new learning tasks. After every fixed set of NELL’s iterations, its growing knowledge base would be input to the relation generation system which will in turn feed NELL with new relation predicates. One additional area for future research is to extend OntExt to discover new categories in addition to new relations.

Acknowledgements

We gratefully acknowledge support for this research from Darpa, Google, Yahoo! and the Brazilian research agency CNPq. We also gratefully acknowledge Dr. Madhavi Ganapathiraju (at University of Pittsburgh) for her support and encouragement.

References

Agichtein, E. and L. Gravano (2000). "Snowball: Extracting relations from large plain-text collections." Procs. of the Fifth ACM International Conference on Digital Libraries.

Banko, M., M. Cararella, et al. (2007). "Open information extraction from the web." In Procs. of IJCAI.

Banko, M. and O. Etzioni (2008). "The Tradeoffs Between Open and Traditional Relation Extraction." In Proceedings of ACL-08.

Callan, J., and Hoy, M. (2009). Clueweb09 data set. http://boston.lti.cs.cmu.edu/Data/clueweb09/.

Carlson, A., J. Betteridge, et al. (2009). "Coupling Semi-Supervised Learning of Categories and Relations." Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing.

A. Carlson, J. Betteridge, et al. (2010). "Coupled Semi-Supervised Learning for Information Extraction," Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM), 2010.

A. Carlson, J. Betteridge, et al., (2010b). "Toward an Architecture for Never-Ending Language Learning," Proceedings of the Conference on Artificial Intelligence (AAAI), 2010.

Etzioni, O., M. Cafarella, et al. (2005). "Unsupervised named-entity extraction from the web: An experimental study." Artificial Intelligence.

Hasegawa, T., S. Sekine, et al. (2004). "Discovering relations among named entities from large corpora." Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics

Zhang, M., J. Su, et al. (2005). "Discovering Relations between Named Entities from a Large Raw Corpus Using Tree Similarity-based Clustering." IJCNLP 05.

Hasegawa, T., S. Sekine, et al. (2004). "Discovering relations among named entities from large corpora." Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics

Zhang, M., J. Su, et al. (2005). "Discovering Relations between Named Entities from a Large Raw Corpus Using Tree Similarity-based Clustering." IJCNL

Hearst, M. (1992) Automatic Acquisition of Hyponyms from Large Text Corpora. Proc. of the Fourteenth International Conference on Computational Linguistics, Nantes, F

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

Structured Relation Discovery using Generative Models

Limin Yao[*] **Aria Haghighi**[+] **Sebastian Riedel**[*] **Andrew McCallum**[*]
[*] Department of Computer Science, University of Massachusetts at Amherst
[+] CSAIL, Massachusetts Institute of Technology
{lmyao,riedel,mccallum}@cs.umass.edu
{aria42}@csail.mit.edu

Abstract

We explore unsupervised approaches to relation extraction between two named entities; for instance, the semantic *bornIn* relation between a person and location entity. Concretely, we propose a series of generative probabilistic models, broadly similar to topic models, each which generates a corpus of observed triples of entity mention pairs and the surface syntactic dependency path between them. The output of each model is a clustering of observed relation tuples and their associated textual expressions to underlying semantic relation types. Our proposed models exploit entity type constraints within a relation as well as features on the dependency path between entity mentions. We examine effectiveness of our approach via multiple evaluations and demonstrate 12% error reduction in precision over a state-of-the-art weakly supervised baseline.

1 Introduction

Many NLP applications would benefit from large knowledge bases of relational information about entities. For instance, knowing that the entity *Steve Balmer* bears the *leaderOf* relation to the entity *Microsoft*, would facilitate question answering (Ravichandran and Hovy, 2002), data mining, and a host of other end-user applications. Due to these many potential applications, relation extraction has gained much attention in information extraction (Kambhatla, 2004; Culotta and Sorensen, 2004; Mintz et al., 2009; Riedel et al., 2010; Yao et al., 2010). We propose a series of generative probabilistic models, broadly similar to standard topic models, which generate a corpus of observed triples of entity mention pairs and the surface syntactic dependency path between them. Our proposed models exploit entity type constraints within a relation as well as features on the dependency path between entity mentions. The output of our approach is a clustering over observed relation paths (e.g. "X was born in Y" and "X is from Y") such that expressions in the same cluster bear the same semantic relation type between entities.

Past work has shown that standard supervised techniques can yield high-performance relation detection when abundant labeled data exists for a fixed inventory of individual relation types (e.g. *leaderOf*) (Kambhatla, 2004; Culotta and Sorensen, 2004; Roth and tau Yih, 2002). However, less explored are *open-domain* approaches where the set of possible relation types are not fixed and little to no labeled is given for each relation type (Banko et al., 2007; Banko and Etzioni, 2008). A more related line of research has explored inducing relation types via clustering. For example, DIRT (Lin and Pantel, 2001) aims to discover different representations of the same semantic relation using distributional similarity of dependency paths. Poon and Domingos (2008) present an Unsupervised semantic parsing (USP) approach to partition dependency trees into meaningful fragments (or "parts" to use their terminology). The combinatorial nature of this dependency partition model makes it difficult for USP to scale to large data sets despite several necessary approximations during learning and infer-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1456–1466,
Edinburgh, Scotland, UK, July 27–31, 2011.

ence. Our work is similar to DIRT and USP in that we induce relation types from observed dependency paths, but our approach is a straightforward and principled generative model which can be efficiently learned. As we show empirically, our approach outperforms these related works when trained with the same amount of data and further gains are observed when trained with more data.

We evaluate our approach using 'intrinsic' clustering evaluation and 'extrinsic' evaluation settings.[1] The former evaluation is performed using subset of induced clusters against Freebase relations, a large manually-built entity and relational database. We also show some clusters which are not included as Freebase relations, as well as some entity clusters found by our approach. The latter evaluation uses the clustering induced by our models as features for relation extraction in distant supervision framework. Empirical results show that we can find coherent clusters. In relation extraction, we can achieve 12% error reduction in precision over a state-of-the-art weakly supervised baseline and we show that using features from our proposed models can find more facts for a relation without significant accuracy loss.

2 Problem and Experimental Setup

The task of relation extraction is mapping surface textual relations to underlying semantic relations. For instance, the textual expression "X was born in Y" indicates a semantic relation *bornIn* between entities "X" and "Y". This relation can be expressed textually in several ways: for instance, "X, a native of Y" or "X grew up in Y". There are several components to a coherent relation type, including a tight small number of textual expressions as well as constraints on the entities involved in the relation. For instance, in the *bornIn* relation "X" must be a person entity and "Y" a location (typically a city or nation). In this work, we present an unsupervised probabilistic generative model for inducing clusters of relation types and recognizing their textual expressions. The set of relation types is not pre-specified but induced from observed unlabeled data. See Table 4 for examples of learned semantic relations.

Our observed data consists of a corpus of documents and each document is represented by a bag of relation tuples. Each tuple represents an observed syntactic relationship between two Named Entities (NE) and consists of three components: the dependency path between two NE mentions, the source argument NE, and the destination argument NE. A dependency path is a concatenation of dependency relations (edges) and words (nodes) along a path in a dependency tree. For instance, the sentence "John Lennnon was born in Liverpool" would yield the relation tuple (*Lennon*, $[\uparrow -nsubjpass, born, \downarrow -in]$, *Liverpool*). This relation tuple reflects a semantic *bornIn* relation between the *John Lennon* and *Liverpool* entities. The dependency path in this example corresponds to the "X was born in Y" textual expression given earlier. Note that for the above example, the *bornIn* relation can only occur between a *person* and a *location*. The relation tuple is the primary observed random variable in our model and we construct our models (see Section 3) so that clusters consist of textual expressions representing the same underlying relation type.

3 Models

We propose three generative models for modeling tuples of entity mention pairs and the syntactic dependency path between them (see Section 2). The first two models, Rel-LDA and Rel-LDA1 are simple extensions of the standard LDA model (Blei et al., 2003). At the document level, our model is identical to standard LDA; a multinomial distribution is drawn over a fixed number of relation types R. Changes lie in the observations. In standard LDA, the atomic observation is a word drawn from a latent topic distribution determined by a latent topic indicator variable for that word position. In our approach, a document consists of an exchangeable set of relation tuples. Each relation tuple is drawn from a relation type 'topic' distribution selected by a latent relation type indicator variable. Relation tuples are generated using a collection of independent features drawn from the underlying relation type distribution. These changes to standard LDA are intended to have the effect that instead of representing semantically related words, the 'topic' latent variable represents a relation type.

Our third model exploits entity type constraints within a relation and induces clusters of relations

[1] See Section 4 for a fuller discussion of evaluation.

and entities jointly. For each tuple, a set of relation level features and two latent entity type indicators are drawn independently from the relation type distribution; a collection of entity mention features for each argument is drawn independently from the entity type distribution selected by the entity type indicator.

Path	X, made by Y
Source	Gamma Knife
Dest	Elekta
Trigger	make
Lex	, made by the Swedish medical technology firm
POS	, VBN IN DT JJ JJ NN NN
NER pair	MISC-ORG
Sync pair	partmod-pobj

Table 1: The features of tuple '(Gamma Knife, made by, Elekta)' in sentence "Gamma Knife, made by the Swedish medical technology firm Elekta, focuses low-dosage gamma radiation ..."

$\|R\|$	Number of relations
$\|D\|$	Number of documents
r	A relation
doc	A document
p, s, d	Dep path, source and dest args
f	A feature/feature type
T	Entity type of one argument
α	Dirichlet prior for θ_{doc}
β_x	Dirichlet prior for ϕ_{rx}
β	Dirichlet prior for ϕ_t
θ_{doc}	$p(r\|doc)$
ϕ_{rx}	$p(x\|r)$
ϕ_t	$p(f_s\|T)$, $p(f_d\|T)$

Table 2: The notation used in our models

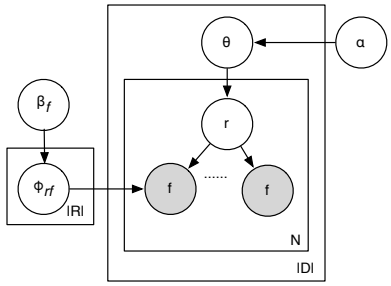


Figure 1: Rel-LDA model. Shaded circles are observations, and unshaded ones are hidden variables. A document consists of N tuples. Each tuple has a set of features. Each feature of a tuple is generated independently from a hidden relation variable r.

3.1 Rel-LDA Model

This model is an extension to the standard LDA model. At the document level, a multinomial distribution over relations θ_{doc} is drawn from a prior Dir(α). To generate a relation tuple, we first draw a relation 'topic' r from Multi(θ). Then we generate each feature f of a tuple independently from a multinomial distribution Multi(ϕ_{rf}) selected by r. In this model, each tuple has three features, i.e. its three components, shown in the first three rows in Table 1. Figure 1 shows the graphical representation of Rel-LDA. Table 2 lists all the notation used in describing our models.

The learning process of the models is an EM process. The procedure is similar to that used by the standard topic model. In the variational E-step (inference), we sample the relation type indicator for each tuple using $p(r|f)$:

$$P(r|f(p,s,d)) \propto p(r)\prod_f p(f|r)$$
$$\propto (\alpha_r + n_{r|d})\prod_f \frac{\beta_f + n_{f|r}}{\sum_{f'}(\beta_{f'} + n_{f'|r})}$$

$p(r)$ and $p(f|r)$ are estimated in the M-step:

$$\theta_{doc} = \frac{\alpha + n_{r|doc}}{\sum_{r'}(\alpha + n_{r'|doc})}$$
$$\phi_{rf} = \frac{\beta_f + n_{f|r}}{\sum_{f'}(\beta_{f'} + n_{f'|r})}$$

where $n_{f|r}$ indicates the number of times a feature f is assigned with r.

3.2 Rel-LDA1

Looking at results of Rel-LDA, we find the clusters sometimes are in need of refinement, and we can address this by adding more features. For instance, adding *trigger* features can encourage sparsity over dependency paths. We define trigger words as all the words on the dependency path except stop words. For example, from path "X, based in Y", "base" is extracted as a trigger word. The intuition

for using trigger words is that paths sharing the same set of trigger words should go to one cluster. Adding named entity tag pair can refine the cluster too. For example, a cluster found by Rel-LDA contains "X was born in Y" and "X lives in Y"; but it also contains "X, a company in Y". In this scenario, adding features 'PER-LOC' and 'ORG-LOC' can push the model to split the clusters into two and put the third case into a new cluster.

Hence we propose Rel-LDA1. It is similar to Rel-LDA, except that each tuple is represented with more features. Besides p, s, and d, we introduce trigger words, lexical pattern, POS tag pattern, the named entity pair and the syntactic category pair features for each tuple. Lexical pattern is the word sequence between the two arguments of a tuple and POS tag pattern is the POS tag sequence of the lexical pattern. See Table 1 as an example.

Following typical EM learning(Charniak and Elsner, 2009), we start with a much simpler generative model, expose the model to fewer features first, and iteratively add more features. First, we train a Rel-LDA model, i.e. the model only generates the dependency path, source and destination arguments. After each interval of 10 iterations, we introduce one additional feature. We add the features in the order of trigger, lexical pattern, POS, NER pair, and syntactic pair.

3.3 Type-LDA model

We know that relations can only hold between certain entity types, known as selectional preferences (Ritter et al., 2010; Seaghdha, 2010; Kozareva and Hovy, 2010). Hence we propose Type-LDA model. This model can capture the selectional preferences of relations to their arguments. In the mean time, it clusters tuples into relational clusters, and arguments into different entity clusters. The entity clusters could be interesting in many ways, for example, defining fine-grained entity types and finding new concepts.

We split the features of a tuple into relation level features and entity level features. Relation level features include the dependency path, trigger, lex and POS features; entity level features include the entity mention itself and its named entity tag.

The generative storyline is as follows. At the document level, a multinomial distribution over relations θ_{doc} is drawn from a Dirichlet prior. A document consists of N relation tuples. Each tuple is represented by relation level features (f) and entity level features of source argument (f_s) and destination argument (f_d). For each tuple, a relation r is drawn from Multi(θ_{doc}). The relation level features and two hidden entity types T_1 and T_2 are independently generated from r. Features f_s are generated from T_1 and f_d from T_2. Figure 2 shows the graphical representation of this model.

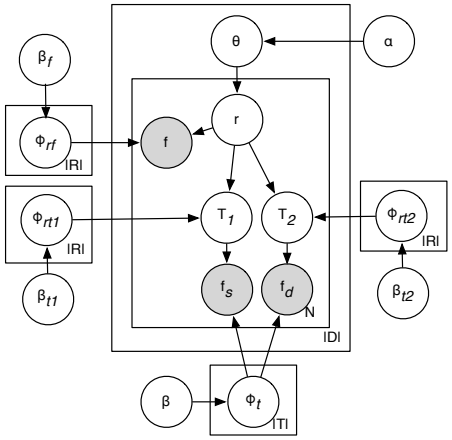


Figure 2: Type-LDA model. Each document consists of N tuples. Each tuple has a set of features, relation level features f and entity level features of source argument f_s and destination argument f_d. Relation level features and two hidden entity types T_1 and T_2 are generated from hidden relation variable r independently. Source entity features are generated from T_1 and destination features are generated from T_2.

At inference time, we sample r, T_1 and T_2 for each tuple. For efficient inference, we first initialize the model without T_1 and T_2, i.e. all the features are generated directly from r. Here the model degenerates to Rel-LDA1. After some iterations, we introduce T_1 and T_2. We sample the relation variable (r) and two mention types variables (T_1,T_2) iteratively for each tuple. We can sample them together, but this is not very efficient. In addition, we found that it does not improve performance.

4 Experiments

Our experiments are carried out on New York Times articles from year 2000 to 2007 (Sandhaus, 2008). We filter out some noisy documents, for example,

obituary content, lists and so on. Obituary articles often contain syntax that diverges from standard newswire text. This leads to parse errors with WSJ-trained parsers and in turn, makes extraction harder. We also filter out documents that contain lists or tables of items (such as books, movies) because this semi-structured information is not the focus of our current work. After filtering we are left with approximately 428K documents. They are preprocessed in several steps. First we employ Stanford tools to tokenize, sentence-split and Part-Of-Speech tag (Toutanova et al., 2003) a document. Next we recognize named entities (Finkel et al., 2005) by labelling tokens with PERSON, ORGANIZATION, LOCATION, MISC and NONE tags. Consecutive tokens which share the same category are assembled into entity mentions. They serve as source and destination arguments of the tuples we seek to model. Finally we parse each sentence of a document using MaltParser (Nivre et al., 2004) and extract dependency paths for each pair of named entity mentions in one sentence.

Following DIRT (Lin and Pantel, 2001), we filter out tuples that do not satisfy the following constraints. First, the path needs to be shorter than 10 edges, since longer paths occur less frequently. Second, the dependency relations in the path should connect two content words, i.e. nouns, verbs, adjectives and adverbs. For example, in phrase 'solve a problem', 'obj(solve, problem)' is kept, while 'det(problem, a)' is discarded. Finally, the dependency labels on the path must not be: 'conj', 'ccomp', 'parataxis', 'xcomp', 'pcomp', 'advcl', 'punct', and 'infmod'. This selection is based on the observation that most of the times the corresponding dependency relations do not explicitly state a relation between two candidate arguments.

After all entity mentions are generated and paths are extracted, we have nearly 2.5M tuples. After clustering (inference), each of these tuple will belong to one cluster/relation and is associated with its clusterID.

We experimented with the number of clusters and find that in a range of 50-200 the performance does not vary significantly with different numbers. In our experiments, we cluster the tuples into 100 relation clusters for all three models. For Type-LDA model, we use 50 entity clusters.

We evaluate our models in two ways. The first aims at measuring the clustering quality by mapping clusters to Freebase relations. The second seeks to assess the utility of our predicted clusters as features for relation extraction.

4.1 Relations discovered by different models

Looking closely at the clusters we predict, we find that some of them can be mapped to Freebase relations. We discover clusters that roughly correspond to the *parentCom* (parent company relation), *filmDirector*, *authorOf*, *comBase* (base of a company relation) and *dieIn* relations in Freebase. We treat Freebase annotations as ground truth and measure recall. We count each tuple in a cluster as true positive if Freebase states the corresponding relation between its argument pair. We find that precision numbers against Freebase are low, below 10%. However, these numbers are not reliable mainly because many correct instances found by our models are missing in Freebase. One reason why our predictions are missing in Freebase is coreference. For example, we predict *parentCom* relation between 'Linksys' and 'Cisco', while Freebase only considers 'Cisco Systems, Inc.' as the parent company of 'Linksys'. It does not corefer 'Cisco' to 'Cisco Systems, Inc.'. Incorporating coreference in our model may fix this problem and is a focus of future work. Instead of measuring precision against Freebase, we ask humans to label 50 instances for each cluster and report precision according to this annotated data. Table 3 shows the scores.

We can see that in most cases Rel-LDA1 and Type-LDA substantially outperform the Rel-LDA model. This is due to the fact that both models can exploit more features to make clustering decisions. For example, in Rel-LDA1 model, the NER pair feature restricts the entity types the two arguments can take.

In the following, we take *parentCom* relation as an example to analyze the behaviors of different models. Rel-LDA includes spurious instances such as 'A is the chief executive of B', while Rel-LDA1 has fewer such instances due to the NER pair feature. Similarly, by explicitly modeling entity type constraints, Type-LDA makes fewer such errors. All our models make mistakes when sentences have coordination structures on which the parser has failed.

Rel.	Sys.	Rec.	Prec.
parentCom	Rel-LDA	51.4	76.0
	Rel-LDA1	49.5	**78.0**
	Type-LDA	**55.3**	72.0
filmDirector	Rel-LDA	42.5	32.0
	Rel-LDA1	70.5	**40.0**
	Type-LDA	**74.2**	26.0
comBase	Rel-LDA	31.5	12.0
	Rel-LDA1	54.2	22.0
	Type-LDA	**57.1**	**30.0**
authorOf	Rel-LDA	25.2	**84.0**
	Rel-LDA1	**46.9**	86.0
	Type-LDA	20.2	68.0
dieIn	Rel-LDA	26.5	34.0
	Rel-LDA1	**55.9**	**40.0**
	Type-LDA	50.2	28.0

Table 3: Clustering quality evaluation (%), Rec. is measured against Freebase, Prec. is measured according to human annotators

For example, when a sentence has the following pattern "The winners are A, a part of B; C, a part of D; E, a part of F", our models may predict *parentCom*(A,F), because the parser connects A with F via the pattern 'a part of'.

Some clusters found by our models cannot be mapped to Freebase relations. Consider the Freebase relation *worksFor* as one example. This relation subsumes all types of employment relationships, irrespective of the role the employee plays for the employer. By contrast, our models discover clusters such as *leaderOf*, *editorOf* that correspond to more specific roles an employee can have. We show some example relations in Table 4. In the table, the 2nd row shows a cluster of employees of news media companies; the 3rd row shows leaders of companies; the last one shows birth and death places of persons. We can see that the last cluster is noisy since we do not handle antonyms in our models. The arguments of the clusters have noise too. For example, 'New York' occurs as a destination argument in the 2nd cluster. This is because 'New York' has high frequency in the corpus and it brings noise to the clustering results. In Table 5 some entity clusters found by Type-LDA are shown. We find different types of companies, such as financial companies and news companies. We also find subclasses of *person*, for example, *reviewer* and *politician*, because these different entity classes participate in different relations. The last cluster shown in the table is a mixture of news companies and government agencies. This may be because this entity cluster is affected by many relations.

4.2 Distant Supervision based Relation Extraction

Our generative models detect clusters of dependency paths and their arguments. Such clusters are interesting in their own right, but we claim that they can also be used to help a supervised relation extractor. We validate this hypothesis in the context of relation extraction with distant supervision using predicted clusters as features.

Following previous work (Mintz et al., 2009), we use Freebase as our distant supervision source, and align related entity pairs to the New York Times articles discussed earlier. Our training and test instances are pairs of entities for which both arguments appear in at least one sentence together. Features of each instance are extracted from all sentences in which both entities appear together. The gold label for each instance comes from Freebase. If a pair of entities is not related according to Freebase, we consider it a negative example. Note that this tends to create some amount of noise: some pairs may be related, but their relationships are not yet covered in Freebase.

After filtering out relations with fewer than 10 instances we have 65 relations and an additional "O" label for unrelated pairs of entities. We call related instances positive examples and unrelated instances negative examples.

We train supervised classifiers using maximum entropy. The baseline classifier employs features that Mintz et al. (2009) used. To extract features from the generative models we proceed as follows. For each pair of entities, we collect all tuples associated with it. For each of these tuples we extract its clusterID, and use this ID as a binary feature.

The baseline system without generative model features is called *Distant*. The classifiers with additional features from generative models are named after the generative models. Thus we have Rel-LDA, Rel-LDA1 and Type-LDA classifiers. We compare

Source	New York, Euro RSCG Worldwide, BBDO Worldwide, American, DDB Worldwide
Path	X, a part of Y; X, a unit of Y; X unit of Y; X, a division of Y; X is a part of Y
Dest	Omnicom Group, Interpublic Group of Companies, WPP Group, Publicis Groupe
Source	Supreme Court, Anna Wintour, William Kristol, Bill Keller, Charles McGrath
Path	X, an editor of Y; X, a publisher of Y; X, an editor at Y; X, an editor in chief of Y; X is an editor of Y;
Dest	The Times, The New York Times, Vogue, Vanity Fair, New York
Source	Kenneth L. Lay, L. Dennis Kozlowski, Bernard J. Ebbers, Thomas R. Suozzi, Bill Gates
Path	X, the executive of Y; X, Y's executive; X, Y executive; X, the chairman of Y; X, Y's chairman
Dest	Enron, Microsoft, WorldCom, Citigroup, Nassau County
Source	Paul J. Browne, John McArdle, Tom Cocola, Claire Buchan, Steve Schmidt
Path	X, a spokesman for Y; X, a spokeswoman for Y; X, Y spokesman; X, Y spokeswoman; X, a commissioner of Y
Dest	White House, Justice Department, Pentagon, United States, State Department
Source	United Nations, Microsoft, Intel, Internet, M. D. Anderson
Path	X, based in Y; X, which is based in Y; X, a company in Y; X, a company based in Y; X, a consultant in Y
Dest	New York, Washington, Manhattan, Chicago, London
Source	Army, Shiite, Navy, John, David
Path	X was born in Y; X die at home in Y; X die in Y; X, son of Y; X die at Y
Dest	Manhattan, World War II, Brooklyn, Los Angeles, New York

Table 4: The path, source and destination arguments of some relations found by Rel-LDA1.

Company	Microsoft, Enron, NBC, CBS, Disney
FinanceCom	Merrill Lynch, Morgan Stanley, Goldman Sachs, Lehman Brothers, Credit Suisse First Boston
News	Notebook, New Yorker, Vogue, Vanity Fair, Newsweek
SportsTeam	Yankees, Mets, Giants, Knicks, Jets
University	University of California, Harvard, Columbia University, New York University, University of Penn.
Art Reviewer	Stephen Holden, Ken Johnson, Roberta Smith, Anthony Tommasini, Grace Glueck
Games	World Series, Olympic, World Cup, Super Bowl, Olympics
Politician	Eliot Spitzer, Ari Fleischer, Kofi Annan, Scott McClellan, Karl Rove
Gov. Agency	Congress, European Union, NATO, Federal Reserve, United States Court of Appeals
News/Agency	The New York Times, The Times, Supreme Court, Security Council, Book Review

Table 5: The entity clusters found by Type-LDA

these against Distant and the DIRT database. For the latter we parse our data using Minipar (Lin, 1998) and extract dependency paths between pairs of named entity mentions. For each path, the top 3 similar paths are extracted from DIRT database. The Minipar path and the similar paths are used as additional features.

For held-out evaluation, we construct the training data from half of the positive examples and half of the negative examples. The remaining examples are used as test data. Note that the number of negative instances is more than 10 times larger than the number of positive instances. At test time, we rank the predictions by the conditional probabilities obtained from the Maximum Entropy classifier. We report precision of top ranked 50 instances for each relation in table 6. From the table we can see that all systems using additional features outperform the Distant system. In average, our best model achieves 4.1% improvement over the distant supervision baseline, 12% error reduction. The precision of *bornIn* is low because in most cases we predict *bornIn* instances as *liveIn*.

We expect systems using generative model features to have higher recall than the baseline. This is difficult to measure, but precision in the high recall area is a signal. We look at top ranked 1000 instances of each system and show the precision in the last row of the table. We can see that our best model Type-LDA outperforms the distant supervision baseline by 4.5%.

Why do generative model features help to im-

Relation	Dist	Rel	Rel1	Type	DIRT
worksFor	80.0	**92.0**	86.0	90.0	84.0
authorOf	98.0	98.0	98.0	98.0	98.0
containedBy	92.0	**96.0**	**96.0**	92.0	**96.0**
bornIn	16.0	18.0	22.0	**24.0**	10.0
dieIn	28.0	**30.0**	28.0	24.0	24.0
liveIn	50.0	52.0	54.0	54.0	**56.0**
nationality	92.0	**94.0**	90.0	90.0	**94.0**
parentCom	94.0	**96.0**	**96.0**	**96.0**	90.0
founder	65.2	**76.3**	61.2	64.0	68.3
parent	52.0	**54.0**	50.0	52.0	52.0
filmDirector	54.0	60.0	60.0	**64.0**	62.0
Avg	65.6	**69.7**	67.4	68.0	66.8
Prec@1K	82.8	85.8	85.3	**87.3**	82.8

Table 6: Precision (%) of some frequent relations

prove relation extraction? One reason is that generative models can transfer information from known patterns to unseen patterns. For example, given "Sidney Mintz, the great food anthropologist at Johns Hopkins University", we want to predict the relation between 'Sidney Mintz' and 'Johns Hopkins University'. The distant supervision system incorrectly predicts the pair as 'O' since it has not seen the path 'X, the anthropologist at Y' in the training data. By contrast, Rel-LDA can predict this pair correctly as *worksFor* because the dependency path of this pair is in a cluster which contains the path 'X, a professor at Y'.

In addition to held-out evaluation we also carry out manual evaluation. To this end, we use all the positive examples and randomly select five times the number of positive examples as negative examples to train a classifier. The remaining negative examples are candidate instances. We rank the predicted instances according to their classification scores. For each relation, we ask human annotators to judge its top ranked 50 instances.

Table 7 lists the manual evaluation results for some frequent relations. We also list how many instances are found for each relation. For almost all the relations, systems using generative model features find more instances. In terms of precision, our models perform comparatively to the baseline, even better for some relations.

We also notice that clustering quality is not consistent with distant supervision performance. Rel-LDA1 can find better clusters than Rel-LDA but it has lower precision in held-out evaluation. Type-LDA underperforms Rel-LDA in average precision but it gets higher precision in a higher recall area, i.e. precision at 1K. One possible reason for the inconsistency is that the baseline distant supervision system already employs features that are used in Rel-LDA1. Another reason may be that the clusters do not overlap with Freebase relations very well, see section 4.1.

4.3 Comparing against USP

We also try to compare against USP (Poon and Domingos, 2008). Due to memory requirements of USP, we are only able to run it on a smaller data set consisting of 1,000 NYT documents; this is three times the amount of data Poon and Domingos (2008) used to train USP.[2] For distant supervision based relation extraction, we only match about 500 Freebase instances to this small data set.

USP provides a parse tree for each sentence and for each mention pair we can extract a path from the tree. Since USP provides clusters of words and phrases, we use the USP clusterID associated with the words on the path as binary features in the classifier.

All models are less accurate when trained on this smaller dataset; we can do as well as USP does, even a little better. USP achieves 8.6% in F1, Rel-LDA 8.7%, Rel-LDA1 10.3%, Type-LDA 8.9% and Distant 10.3%. Of course, given larger datasets, the performance of Rel-LDA, Rel-LDA1, and Type-LDA improves considerably. In summary, comparing against USP, our approach scales much more easily to large data.

5 Related Work

Many approaches have been explored in relation extraction, including bootstrapping, supervised classification, distant supervision, and unsupervised approaches.

Bootstrapping employs a few labeled examples for each relation, iteratively extracts patterns from the labeled seeds, and uses the patterns to extract

[2]Using the publicly released USP code, training a model with 1,000 documents resulted in about 45 gigabytes of heap space in the JVM.

Relation	Top 50 (%)			#Instances		
	Dist	Rel	Type	Dist	Rel	Type
worksFor	**100.0**	**100.0**	**100.0**	314	**349**	**349**
authorOf	94.0	94.0	**96.0**	185	208	**229**
containedBy	**98.0**	**98.0**	**98.0**	670	714	**804**
bornIn	82.6	**88.2**	88.0	46	36	**56**
dieIn	**100.0**	**100.0**	**100.0**	167	176	**231**
liveIn	**98.0**	**98.0**	94.0	77	86	**109**
nationality	78.0	**82.0**	76.0	84	92	**114**
parentCom	79.2	77.4	**85.7**	24	**31**	28
founder	**80.0**	**80.0**	50.0	5	5	**14**
parent	**97.0**	92.3	94.7	33	**39**	38
filmDirector	92.6	96.9	**97.1**	27	32	**34**

Table 7: Manual evaluation, Precision and recall of some frequent relations

more seeds (Brin, 1998). This approach may suffer from low recall since the patterns can be too specific.

Supervised learning can discover more general patterns (Kambhatla, 2004; Culotta and Sorensen, 2004). However, this approach requires labeled data, and most work only carry out experiments on small data set.

Distant supervision for relation extraction requires no labeled data. The approach takes some existing knowledge base as supervision source, matches its relational instances against the text corpus to build the training data, and extracts new instances using the trained classifiers (Mintz et al., 2009; Bunescu and Mooney, 2007; Riedel et al., 2010; Yao et al., 2010).

All these approaches can not discover new relations and classify instances which do not belong to any of the predefined relations. Other past work has explored inducing relations using unsupervised approaches.

For example, DIRT (Lin and Pantel, 2001) aims to discover different representations of the same semantic relation, i.e. similar dependency paths. They employ the distributional similarity based approach while we use generative models. Both DIRT and our approach take advantage of the arguments of dependency paths to find semantic relations. Moreover, our approach can cluster the arguments into different types.

Unsupervised semantic parsing (USP) (Poon and Domingos, 2008) discovers relations by merging predicates which have similar meanings; it proceeds to recursively cluster dependency tree fragments (or "parts") to best explain the observed sentence. It is not focused on capturing any particular kind of relation between sentence constituents, but to capture repeated patterns. Our approach differs in that we are focused on capturing a narrow range of binary relations between named entities; some of our models (see Section 3) utilize entity type information to constraint relation type induction. Also, our models are built to be scalable and trained on a very large corpus. In addition, we use a distant supervision framework for evaluation.

Relation duality (Bollegala et al., 2010) employs co-clustering to find clusters of entity pairs and patterns. They identify each cluster of entity pairs as a relation by selecting representative patterns for that relation. This approach is related to our models, however, it does not identify any entity clusters.

Generative probabilistic models are widely employed in relation extraction. For example, they are used for in-domain relation discovery while incorporating constraints via posterior regularization (Chen et al., 2011). We are focusing on open domain relation discovery. Generative models are also applied to selectional preference discovery (Ritter et al., 2010; Seaghdha, 2010). In this scenario, the authors assume relation labels are given while we automatically discover relations. Generative models are also used in unsupervised coreference (Haghighi and Klein, 2010).

Clustering is also employed in relation extraction. Hasegawa et al. (2004) cluster pairs of named entities according to the similarity of context words intervening between them. Their approach is not probabilistic. Researchers also use topic models to perform dimension reduction on features when they cluster relations (Hachey, 2009). However, they do not explicitly model entity types.

Open information extraction aims to discover relations independent of specific domains and relations (Banko et al., 2007; Banko and Etzioni, 2008). A self-learner is employed to extract relation instances but the systems do not cluster the instances into relations. Yates and Etzioni (2009) present RESOLVER for discovering relational synonyms as a post processing step. Our approach integrates entity and relation discovery in a probabilistic model.

6 Conclusion

We have presented an unsupervised probabilistic generative approach to relation extraction between two named entities. Our proposed models exploit entity type constraints within a relation as well as features on the dependency path between entity mentions to cluster equivalent textual expressions. We demonstrate the effectiveness of this approach by comparing induced relation clusters against a large knowledge base. We also show that using clusters of our models as features in distant supervised framework yields 12% error reduction in precision over a weakly supervised baseline and outperforms other state-of-the art relation extraction techniques.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and the University of Massachusetts gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181, ITR#1, and NSF MALLET. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI2007*.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2010. Relational duality: Unsupervised extraction of semantic relations between entities on the web. In *Proceedings of WWW*.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *Proc. of WebDB Workshop at 6th International Conference on Extending Database Technology*.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45rd Annual Meeting of the Association for Computational Linguistics (ACL '07)*.

Eugene Charniak and Micha Elsner. 2009. Em works for pronoun anaphora resolution. In *Proceedings of ACL*.

Harr Chen, Edward Benson, Tahira Naseem, and Regina Barzilay. 2011. In-domain relation discovery with meta-constraints via posterior regularization. In *Proceedings of ACL*.

Aron Culotta and Jeffery Sorensen. 2004. Dependency tree kernels for relation extraction. In *42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 363–370, June.

Benjamin Hachey. 2009. *Towards Generic Relation Extraction*. Ph.D. thesis, University of Edinburgh.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Proceedings of HLT-NAACL*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *ACL*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACL*.

Zornitsa Kozareva and Eduard Hovy. 2010. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of ACL 10*.

Dekang Lin and Patrick Pantel. 2001. Dirt - discovery of inference rules from text. In *Proceedings of KDD*.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.

Hoifung Poon and Pedro Domingos. 2008. Unsupervised semantic parsing. In *Proceedings of the Conference on Empirical methods in natural language processing (EMNLP)*.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD '10)*.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of ACL10*.

Dan Roth and Wen tau Yih. 2002. Probabilistic reasoning for entity and relation recognition. In *Proceedings of Coling*.

Evan Sandhaus, 2008. *The New York Times Annotated Corpus*. Linguistic Data Consortium, Philadelphia.

Diarmuid O Seaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of ACL 10*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*, pages 252–259.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023, Cambridge, MA, October. Association for Computational Linguistics.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.

Closing the Loop: Fast, Interactive Semi-Supervised Annotation With Queries on Features and Instances

Burr Settles
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
bsettles@cs.cmu.edu

Abstract

This paper describes DUALIST, an active learning annotation paradigm which solicits and learns from labels on both features (e.g., words) and instances (e.g., documents). We present a novel semi-supervised training algorithm developed for this setting, which is (1) fast enough to support real-time interactive speeds, and (2) at least as accurate as preexisting methods for learning with mixed feature and instance labels. Human annotators in user studies were able to produce near-state-of-the-art classifiers—on several corpora in a variety of application domains—with only a few minutes of effort.

1 Introduction

In active learning, a classifier participates in its own training process by posing *queries*, such as requesting labels for documents in a text classification task. The goal is to maximize the accuracy of the trained system in the most economically efficient way. This paradigm is well-motivated for natural language applications, where unlabeled data may be readily available (e.g., text on the Internet), but the annotation process can be slow and expensive.

Nearly all previous work in active learning, however, has focused on selecting queries from the *learner*'s perspective. For example, experiments are often run in simulation rather than with user studies, and results are routinely evaluated in terms of training set size rather than human annotation time or labor costs (which are more reasonable measures of labeling effort). Many state-of-the-art algorithms are also too slow to run or too tedious to implement to be useful for real-time interaction with human annotators, and few analyses have taken these factors into account. Furthermore, there is very little work on actively soliciting domain knowledge from humans (e.g., information about features) and incorporating this into the learning process.

While selecting good queries is clearly important, if our goal is to reduce actual annotation effort these human factors must be taken into account. In this work, we propose a new interactive annotation interface which addresses some of these issues; in particular it has the ability to pose queries on both features (e.g., words) and instances (e.g., documents). We present a novel semi-supervised learning algorithm that is fast, flexible, and accurate enough to support these interface design constraints interactively.

2 DUALIST: Utility for Active Learning with Instances and Semantic Terms

Figure 1 shows a screenshot of the DUALIST annotation tool, which is freely available as an open-source software project[1]. On the left panel, users are presented with unlabeled documents: in this case Usenet messages that belong to one of two sports-related topics: *baseball* and *hockey*. Users may label documents by clicking on the class buttons listed below each text. In cases of extreme ambiguity, users may ignore a document by clicking the "X" to remove it from the pool of possible queries.

On the right panel, users are given a list of feature queries organized into columns by class label.

[1] http://code.google.com/p/dualist/

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1467–1478,
Edinburgh, Scotland, UK, July 27–31, 2011.

Figure 1: A screenshot of DUALIST.

The rationale for these columns is that they should reduce cognitive load (i.e., once a user is in the *baseball* mindset, s/he can simply go down the list, labeling features in context: "plate," "pitcher," "bases," etc.). Within each column, words are sorted by how informative they are the to classifier, and users may click on words to label them. Each column also contains a text box, where users may "inject" domain knowledge by typing in arbitrary words (whether they appear in any of the columns or not). The list of previously labeled words appears at the bottom of each list (highlighted), and can be unlabeled at any time, if users later feel they made any errors.

Finally, a large submit button is located at the top of the screen, which users must click to re-train the classifier and receive a new set of queries. The learning algorithm is actually fast enough to do this automatically after each labeling action. However, we found such a dynamically changing interface to be frustrating for users (e.g., words they wanted to label would move or disappear).

2.1 A Generative Model for Learning from Feature and Instance Labels

For the underlying model in this system, we use multinomial naïve Bayes (MNB) since it is simple, fast, and known to work well for several natural language applications—text classification in particular—despite its simplistic and often violated independence assumptions (McCallum and Nigam, 1998; Rennie et al., 2003).

MNB models the distribution of features as a multinomial: documents are sequences of words, with the "naïve" assumption that words in each position are generated independently. Each document is treated as a mixture of classes, which have their own multinomial distributions over words. Let the model be parameterized by the vector θ, with $\theta_j = P(y_j)$ denoting the probability of class y_j, and $\theta_{jk} = P(f_k|y_j)$ denoting the probability of generating word f_k given class y_j. Note that for class priors $\sum_j \theta_j = 1$, and for per-class word multinomials $\sum_k \theta_{jk} = 1$. The likelihood of document x being generated by class y_j is given by:

$$P_\theta(x|y_j) = P(|x|) \prod_k (\theta_{jk})^{f_k(x)},$$

where $f_k(x)$ is the frequency count of word f_k in document x. If we assume $P(|x|)$ is distributed independently of class, and since document length $|x|$ is fixed, we can drop the first term for classification purposes. Then, we can use Bayes' rule to calculate the posterior probability under the model of a label, given the input document for classification:

$$P_\theta(y_j|x) = \frac{P_\theta(y_j)P_\theta(x|y_j)}{P_\theta(x)} = \frac{\theta_j \prod_k (\theta_{jk})^{f_k(x)}}{Z(x)}, \quad (1)$$

where $Z(x)$ is shorthand for a normalization constant, summing over all possible class labels.

The task of training such a classifier involves estimating the parameters in θ, given a set of labeled instances $\mathcal{L} = \{\langle x^{(l)}, y^{(l)} \rangle\}_{l=1}^{L}$. To do this, we use a Dirichlet prior and take the expectation of each parameter with respect to the posterior, which is a simple way to estimate a multinomial (Heckerman, 1995). In other words, we count the fraction of times the word f_k occurs in the labeled set among documents of class y_j, and the prior adds m_{jk} "hallucinated" occurrences for a smoothed version of the maximum likelihood estimate:

$$\theta_{jk} = \frac{m_{jk} + \sum_i P(y_j|x^{(i)}) f_k(x^{(i)})}{Z(f_k)}. \quad (2)$$

Here, m_{jk} is the prior for word f_k under class y_j, $P(y_j|x^{(i)}) \in \{0, 1\}$ indicates the true labeling of the ith document in the training set, and $Z(f_k)$ is a normalization constant summing over all words in the vocabulary. Typically, a uniform prior is used, such as the Laplacian (a value of 1 for all m_{jk}). Class parameters θ_j are estimated a similar way, by counting

the fraction of documents that are labeled with that class, subject to a prior m_j. This prior is important in the event that no documents are yet labeled with y_j, which can be quite common early on in the active learning process.

Recall that our scenario lets human annotators provide not only document labels, but feature labels as well. To make use of this additional information, we assume that labeling the word f_k with a class y_j increases the probability $P(f_k|y_j)$ of the word appearing in documents of that class. The natural interpretation of this under our model is to increase the prior m_{jk} for the corresponding multinomial. To do this we introduce a new parameter α, and define the elements of the Dirichlet prior as follows:

$$m_{jk} = \begin{cases} 1+\alpha & \text{if } f_k \text{ is labeled with } y_j, \\ 1 & \text{otherwise.} \end{cases}$$

This approach is extremely flexible, and offers three particular advantages over the previous "pooling multinomials" approach for incorporating feature labels into MNB (Melville et al., 2009). The pooling multinomials algorithm averages together two sets of θ_{jk} parameters: one that is estimated from labeled data, and another derived from feature labels under the assumption of a boolean output variable (treating labeled features are "polarizing" factors). Therefore, pooling multinomials can only be applied to binary classification tasks, while our method works equally well for problems with multiple classes. The second advantage is that feature labels need not be mutually exclusive, so the word "score" could be labeled with both *baseball* and *hockey*, if necessary (e.g., if the task also includes several non-sports labels). Finally, our framework allows users to conceivably provide feature-specific priors α_{jk} to, for example, imply that the word "inning" is a stronger indicator for *baseball* than the word "score" (which is a more general sports term). However, we leave this aspect for future work and employ the fixed-α approach as described above in this study.

2.2 Exploiting Unlabeled Data

In addition to document and feature labels, we usually have access to a large unlabeled corpus. In fact, these texts form the pool of possible instance queries in active learning. We can take advantage of this additional data in generative models like MNB by employing the Expectation-Maximization (EM) algorithm. Combining EM with pool-based active learning was previously studied in the context of instance labeling (McCallum and Nigam, 1998), and we extend the method to our interactive scenario, which supports feature labeling as well.

First, we estimate initial parameters θ' as in Section 2.1, but using *only* the priors (and no instances). Then, we apply the induced classifier on the unlabeled pool $\mathcal{U} = \{x^{(u)}\}_{u=1}^{U}$ (Eq. 1). This is the "E" step of EM. Next we re-estimate feature multinomials θ_{jk}, using both labeled instances from $\mathcal{L}$ and probabilistically-labeled instances from $\mathcal{U}$ (Eq. 2). In other words, $P(y_j|x) \in \{0,1\}$ for $x \in \mathcal{L}$, and $P(y_j|x) = P_{\theta'}(y_j|x)$ for $x \in \mathcal{U}$. We also weight the data in $\mathcal{U}$ by a factor of 0.1, so as not to overwhelm the training signal coming from true instance labels in $\mathcal{L}$. Class parameters θ_j are re-estimated in the analogous fashion. This is the "M" step.

For speed and interactivity, we actually stop training after this first iteration. When feature labels are available, we found that EM generally converges in four to 10 iterations, requiring more training time but rarely improving accuracy (the largest gains consistently come in the first iteration). Also, we ignore labeled data in the initial estimation of θ' because $\mathcal{L}$ is too small early in active learning to yield good results with EM. Perhaps this can be improved by using an ensemble (McCallum and Nigam, 1998), but that comes at further computational expense. Feature labels, on the other hand, seem generally more reliable for probabilistically labeling $\mathcal{U}$.

2.3 Selecting Instance and Feature Queries

The final algorithmic component to our system is the selection of informative queries (i.e., unlabeled words and documents) to present to the annotator.

Querying instances is the traditional mode of active learning, and is well-studied in the literature; see Settles (2009) for a review. In this work we use entropy-based uncertainty sampling, which ranks all instances in $\mathcal{U}$ by the posterior class entropy under the model $H_\theta(Y|x) = -\sum_j P_\theta(y_j|x) \log P_\theta(y_j|x)$, and asks the user to label the top D unlabeled documents. This simple heuristic is an approximation to querying the instance with the maximum information gain (since the class entropy, once labeled, is zero), under the assumption that each x is repre-

sentative of the underlying natural data distribution. Moreover, it is extremely fast to compute, which is important for our interactive environment.

Querying features, though, is a newer idea with significantly less research behind it. Previous work has either assumed that (1) features are not assigned to classes, but instead flagged for "relevance" to the task (Godbole et al., 2004; Raghavan et al., 2006), or (2) feature queries are posed just like instance queries: a word is presented to the annotator, who must choose among the labels (Druck et al., 2009; Attenberg et al., 2010). Recall from Figure 1 that we want to organize feature queries into columns by class label. This means our active learner must produce queries that are class-specific.

To select these feature queries, we first rank elements in the vocabulary by information gain (IG):

$$IG(f_k) = \sum_{\mathbf{I}_k} \sum_{j} P(\mathbf{I}_k, y_j) \log \frac{P(\mathbf{I}_k, y_j)}{P(\mathbf{I}_k)P(y_j)},$$

where $\mathbf{I}_k \in \{0, 1\}$ is a variable indicating the presence or absence of a feature. This is essentially the common feature-selection method for identifying the most salient features in text classification (Sebastiani, 2002). However, we use both $\mathcal{L}$ and probabilistically-labeled instances from $\mathcal{U}$ to compute $IG(f_k)$, to better reflect what the model believes it has learned. To organize queries into classes, we take the top V unlabeled features and pose f_k for the class y_j with which it occurs most frequently, as well as any other class with which it occurs at least 75% as often. Intuitively, this approach (1) queries features that the model believes are most informative, and (2) automatically identifies classes that seem most correlated. To our knowledge, DUALIST is the first active learning environment with both of these properties.

3 Experiments

We conduct four sets of experiments to evaluate our approach. The first two are "offline" experiments, designed to better understand (1) how our training algorithm compares to existing methods for feature-label learning, and (2) the effects of tuning the α parameter. The other experiments are user studies designed to empirically gauge how well human annotators make use of DUALIST in practice.

We use a variety of benchmark corpora in the following evaluations. Reuters (Rose et al., 2002) is a collection of news articles organized into topics, such as *acquisitions*, *corn*, *earnings*, etc. As in previous work (Raghavan et al., 2006) we use the 10 most frequent topics, but further process the corpus by removing ambiguous documents (i.e., that belong to multiple topics) so that all articles have a unique label, resulting in a corpus of 9,002 articles. WebKB (Craven et al., 1998) consists of 4,199 university web pages of four types: *course*, *faculty*, *project*, and *student*. 20 Newsgroups (Lang, 1995) is a set of 18,828 Usenet messages from 20 different online discussion groups. For certain experiments (such as the one shown in Figure 1), we also use topical subsets. Movie Reviews (Pang et al., 2002) is a set of 2,000 online movie reviews categorized as *positive* or *negative* in sentiment. All data sets were processed using lowercased unigram features, with punctuation and common stop-words removed.

3.1 Comparison of Learning Algorithms

An important question is how well our learning algorithm, "MNB/Priors," performs relative to existing baseline methods for learning with labeled features. We compare against two such approaches from the literature. "MaxEnt/GE" is a maximum entropy classifier trained using generalized expectation (GE) criteria (Druck et al., 2008), which are constraints used in training discriminative linear models. For labeled features, these take the form of expected "reference distributions" conditioned on the presence of the feature (e.g., 95% of documents containing the word "inning" should be labeled *baseball*). For each constraint, a term is added to the objective function to encourage parameter settings that yield predictions conforming to the reference distribution on unlabeled instances. "MNB/Pool" is naïve Bayes trained using the pooling multinomials approach (Melville et al., 2009) mentioned in Section 2.1. We also expand upon MNB/Pool using an EM variant to make it semi-supervised.

We use the implementation of GE training from the open-source MALLET toolkit[2], and implement both MNB variants in the same data-processing pipeline. Because the GE implementation available

[2] http://mallet.cs.umass.edu

Corpus	MaxEnt/GE		MNB/Pool		Pool+EM$_1$		MNB/Priors		Priors+EM$_1$	
Reuters	82.8	(22.9)	–	–	–	–	83.7	(≤0.1)	**86.6**	(0.3)
WebKB	22.2	(4.9)	–	–	–	–	67.5	(≤0.1)	**67.8**	(0.1)
20 Newsgroups	49.7	(326.6)	–	–	–	–	50.1	(0.2)	**70.7**	(6.9)
Science	86.9	(5.7)	–	–	–	–	71.4	(≤0.1)	**92.8**	(0.1)
Autos/Motorcycles	90.8	(0.8)	90.1	(≤0.1)	97.5	(≤0.1)	89.9	(≤0.1)	**97.6**	(≤0.1)
Baseball/Hockey	49.9	(0.8)	90.7	(≤0.1)	96.7	(≤0.1)	90.5	(≤0.1)	**96.9**	(≤0.1)
Mac/PC	50.5	(0.6)	86.7	(≤0.1)	**91.2**	(≤0.1)	86.6	(≤0.1)	90.2	(≤0.1)
Movie Reviews	68.8	(1.8)	68.0	(≤0.1)	**73.4**	(0.1)	67.7	(≤0.1)	72.0	(0.1)

Table 1: Accuracies and training times for different feature-label learning algorithms on benchmark corpora. Classification accuracy is reported for each model, using only the top 10 oracle-ranked features per label (and no labeled instances) for training. The best model for each corpus is highlighted in bold. Training time (in seconds) is shown in parentheses on the right side of each column. All results are averaged across 10 folds using cross-validation.

to us only supports labeled features (and not labeled instances as well), we limit the MNB methods to features for a fair comparison. To obtain feature labels in this experiment, we simulate a "feature oracle" as in previous work (Druck et al., 2008; Attenberg et al., 2010), which is essentially the query selection algorithm from Section 2.3, but using complete labeled data to compute $IG(f_k)$. We conservatively use only the top 10 features per class, which is meant to resemble a handful of very salient features that a human might brainstorm to jumpstart the learning process. We experiment with EM$_1$ (one-step EM) variants of both MNB/Pool and MNB/Priors, and set $\alpha = 50$ for the latter (see Section 3.2 for details on tuning this parameter). Results are averaged over 10 folds using cross-validation, and all experiments are conducted on a single 2.53GHz processor machine.

Results are shown in Table 1. As expected, adding one iteration of EM for semi-supervised training improves the accuracy of both MNB methods across all data sets. These improvements come without significant overhead in terms of time: training still routinely finishes in a fraction of a second per fold. MNB/Pool and MNB/Priors, where they can be compared, perform virtually the same as each other with or without EM, in terms of accuracy and speed alike. However, MNB/Pool is only applicable to binary classification problems. As explained in Section 2.1, MNB/Priors is more flexible, and preferable for a more general-use interactive annotation tool like DUALIST.

The semi-supervised MNB methods are also consistently more accurate than GE training—and are about 40 times faster as well. The gains of Priors+EM$_1$ over MaxEnt/GE are statistically significant in all cases but two: Autos/Motorcycles and Movie Reviews[3]. MNB is superior when using anywhere from five to 20 oracle-ranked features per class, but as the number of feature labels increases beyond 30, GE is often more accurate (results not shown). If we think of MaxEnt/GE as a discriminative analog of MNB/Priors+EM, this is consistent with what is known about labeled set size in supervised learning for generative/discriminative pairs (Ng and Jordan, 2002). However, the time complexity of GE training increases sharply with each new labeled feature, since it adds a new constraint to the objective function whose gradient must be computed using all the unlabeled data. In short, GE training is too slow and too inaccurate early in the active learning process (where labels are more scarce) to be appropriate for our scenario. Thus, we select MNB/Priors to power the DUALIST interface.

3.2 Tuning the Parameter α

A second question is how sensitive the accuracy of MNB/Priors is to the parameter α. To study this, we ran experiments varying α from from one to 2^{12}, using different combinations of labeled instances and/or features (again using the simulated oracle and 10-fold cross-validation).

[3]Paired 2-tailed t-test, $p < 0.05$, correcting for multiple tests using the Bonferroni method.

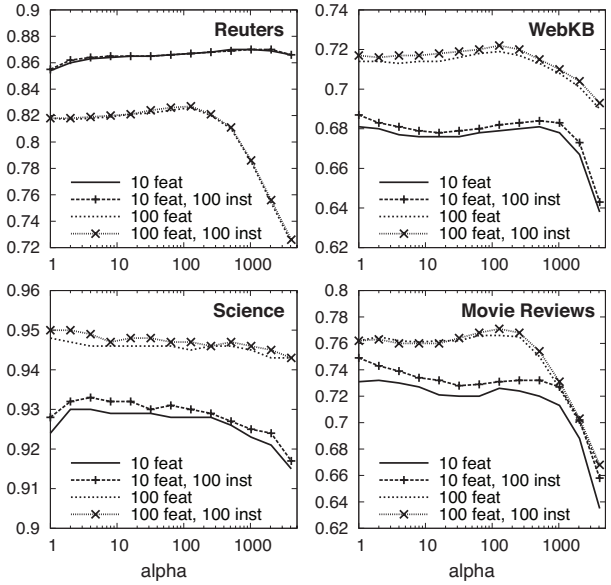


Figure 2: The effects of varying α on accuracy for four corpora, using differing amounts of training data (labeled features and/or instances). For clarity, vertical axes are scaled differently for each data set, and horizontal axes are plotted on a logarithmic scale. Classifier performance remains generally stable across data sets for $\alpha < 100$.

Figure 2 plots these results for four of the corpora. The first thing to note is that in all cases, accuracy is relatively stable for $\alpha < 100$, so tuning this value seems not to be a significant concern; we chose 50 for all other experiments in this paper. A second observation is that, for all but the Reuters corpus, labeling 90 additional features improves accuracy much more than labeling 100 documents. This is encouraging, since labeling features (e.g., words) is known to be generally faster and easier for humans than labeling entire instances (e.g., documents).

For Reuters, however, the additional feature labels appear harmful. The anomaly can be explained in part by previous work with this corpus, which found that a few expertly-chosen keywords can outperform machine learning methods (Cohen and Singer, 1996), or that aggressive feature selection—i.e., using only three or four features per class—helps tremendously (Moulinier, 1996). Corpora like Reuters may naturally lend themselves to feature selection, which is (in some sense) what happens when labeling features. The simulated oracle here was forced to label 100 features, some with very low information gain (e.g., "south" for *acquisitions*, or "proven" for *gold*); we would not expect humans annotators to provide such misleading information. Instead, we hypothesize that in practice there may be a limited set of features with high enough information content for humans to feel confident labeling, after which they switch their attention to labeling instance queries instead. This further indicates that the user-guided flexibility of annotation in DUALIST is an appropriate design choice.

3.3 User Experiments

To evaluate our system in practice, we conducted a series of user experiments. This is in contrast to most previous work, which simulates active learning by using known document labels and feature labels from a simulated oracle (which can be flawed, as we saw in the previous section). We argue that this is an important contribution, as it gives us a better sense of how well the approach actually works in practice. It also allows us to analyze behavioral results, which in turn may help inform future protocols for human interaction in active learning.

DUALIST is implemented as a web-based application in Java and was deployed online. We used three different configurations: *active dual* (as in Figure 1, implementing everything from Section 2), *active instance* (instance queries only, no features), and a *passive instance* baseline (instances only, but selected at random). We also began by randomly selecting instances in the active configurations, until every class has at least one labeled instance or one labeled feature. $D = 2$ documents and $V = 100$ features were selected for each round of active learning.

We recruited five members of our research group to label three data sets using each configuration, in an order of their choosing. Users were first allowed to spend a minute or two familiarizing themselves with DUALIST, but received no training regarding the interface or data sets. All experiments used a fixed 90% train, 10% test split which was consistent across all users, and annotators were not allowed to see the accuracy of the classifier they were training at any time. Each annotation action was timestamped and logged for analysis, and each experiment automatically terminated after six minutes.

Figure 3 shows learning curves, in terms of accuracy vs. annotation time, for each trial in the user study. The first thing to note is that the active

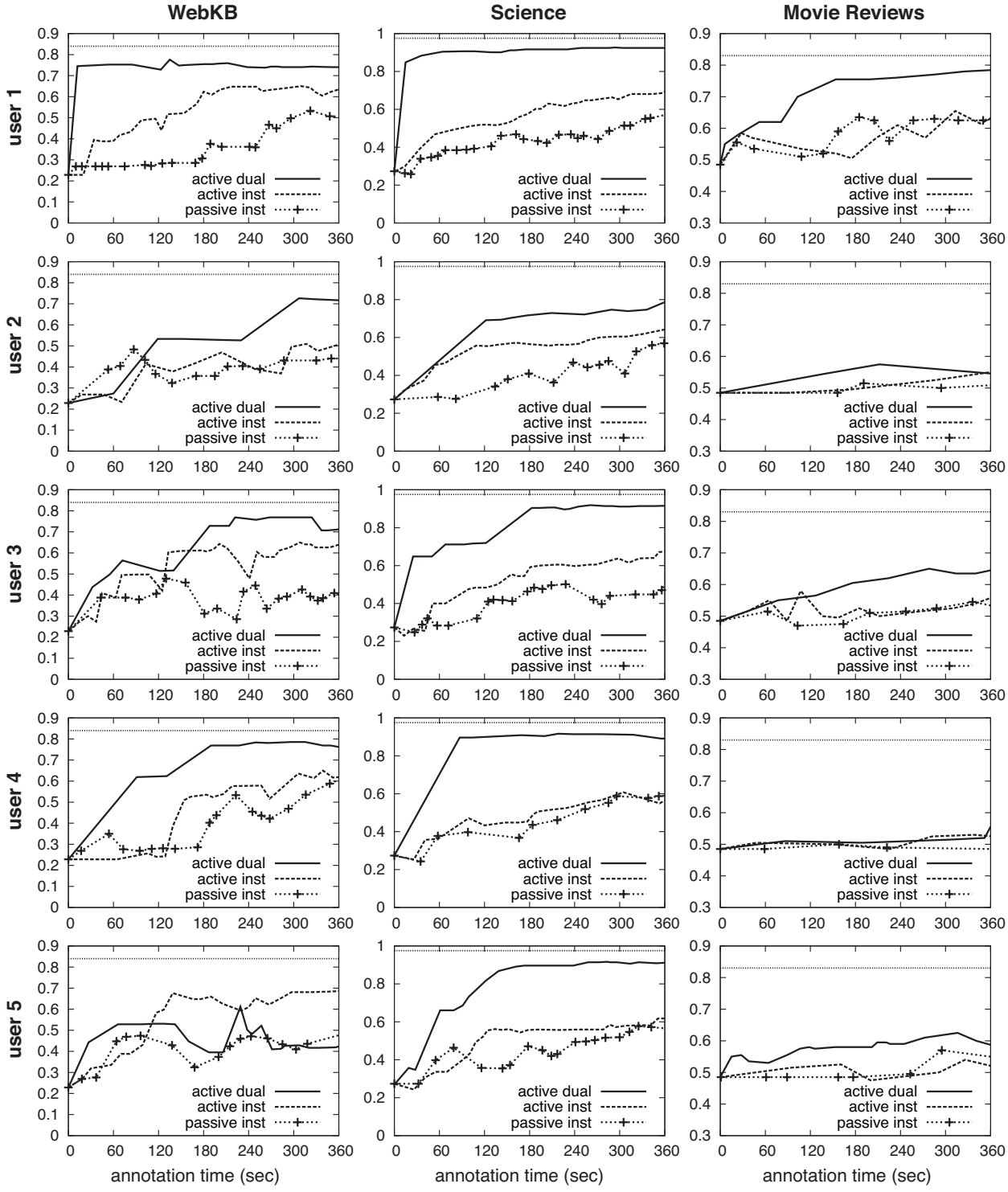


Figure 3: User experiments involving human annotators for text classification. Each row plots accuracy vs. time learning curves for a particular user (under all three experimental conditions) for each of the three corpora (one column per data set). For clarity, vertical axes are scaled differently for each corpus, but held constant across all users. The thin dashed lines at the top of each plot represents the idealized fully-supervised accuracy. Horizontal axes show labeling cost in terms of actual elapsed annotation time (in seconds).

dual configuration yields consistently better learning curves than either active or passive learning with instances alone, often getting within 90% of fully-supervised accuracy (in under six minutes). The only two exceptions make interesting (and different) case studies. User 4 only provided four labeled features in the Movie Review corpus, which partially explains the similarity in performance to the instance-only cases. Moreover, these were manually-added features, i.e., he never answered any of the classifier's feature *queries*, thus depriving the learner of the information it requested. User 5, on the other hand, never manually added features and only answered queries. With the WebKB corpus, however, he apparently found feature queries for the *course* label to be easier than the other classes, and 71% of all his feature labels came from that class (sometimes noisily, e.g., "instructor" might also indicate *faculty* pages). This imbalance ultimately biased the learner toward the *course* label, which led to classification errors. These pathological cases represent potential pitfalls that could be alleviated with additional user studies and training. However, we note that the active dual interface is not particularly worse in these cases, it is simply not significantly better, as in the other 13 trials.

Feature queries were less costly than instances, which is consistent with findings in previous work (Raghavan et al., 2006; Druck et al., 2009). The least expensive actions in these experiments were labeling (mean 3.2 seconds) and unlabeling (1.8s) features, while manually adding new features took only slightly longer (5.9s). The most expensive actions were labeling (10.8s) and ignoring (9.9s) instance queries. Interestingly, we observed that the human annotators spent most of the first three minutes performing feature-labeling actions ( ), and switched to more instance-labeling activity for the final three minutes ( ). As hypothesized in Section 3.2, it seems that the active learner is exhausting the most salient feature queries early on, and users begin to focus on more interpretable instance queries over time. However, more study (and longer annotation periods) are warranted to better understand this phenomenon, which may suggest additional user interface design improvements.

We also saw surprising trends in annotation quality. In active settings, users made an average of one instance-labeling error per trial (relative to the gold-standard labels), but in the passive case this rose to 1.6, suggesting they are more accurate on the active queries. However, they also explicitly ignored more instances in the active dual condition (7.7) than either active instance (5.9) or passive (2.5), indicating that they find these queries more ambiguous. This seems reasonable, since these are the instances the classifier is least certain about. But if we look at the *time* users spent on these actions, they are much faster to label/ignore (9.7s/7.5s) in the active dual scenario than in the active instance (10.0s/10.7s) or passive (12.3s/15.4s) cases, which means they are being more efficient. The differences in time between dual and passive are statistically significant[4].

3.4 Additional Use Cases

Here we discuss the application of DUALIST to a few other natural language processing tasks. This section is not meant to show its superiority relative to other methods, but rather to demonstrate the flexibility and potential of our approach in a variety of problems in human language technology.

3.4.1 Word Sense Disambiguation

Word Sense Disambiguation (WSD) is the problem of determining which meaning of a word is being used in a particular context (e.g., "hard" in the sense of a challenging task vs. a marble floor). We asked a user to employ DUALIST for 10 minutes for each of three benchmark WSD corpora (Mohammad and Pedersen, 2004): Hard (3 senses), Line (6 senses), and Serve (4 senses). Each instance represents a sentence using the ambiguous word, and features are lowercased unigram and bigram terms from the surrounding context in the sentence. The learned models' prediction accuracies (on the sentences *not* labeled by the user) were: 83.0%, 78.4%, and 78.7% for Hard, Line, and Serve (respectively), which appears to be comparable to recent supervised learning results in the WSD literature on these data sets. However, our results were achieved in less than 10 minutes of effort each, by labeling an average of 76 sentences and 32 words or phrases per task (compared to the thousands of labeled training sentences used in previous work).

[4] Kolmogorov-Smirnov test, $p < 0.01$.

3.4.2 Information Extraction

DUALIST is also well-suited to a kind of large-scale information extraction known as semantic class learning: given a set of semantic categories and a very large unlabeled text corpus, learn to populate a knowledge base with words or phrases that belong to each class (Riloff and Jones, 1999; Carlson et al., 2010). For this task, we first processed 500 million English Web pages from the ClueWeb09 corpus (Callan and Hoy, 2009) by using a shallow parser. Then we represented noun phrases (e.g., "Al Gore," "World Trade Organization," "upholstery") as instances, using a vector of their co-occurrences with heuristic contextual patterns (e.g., "visit to X" or "X's mission") as well as a few orthographic patterns (e.g., capitalization, head nouns, affixes) as features. We filtered out instances or contexts that occurred fewer than 200 times in the corpus, resulting in 49,923 noun phrases and 87,760 features.

We then had a user annotate phrases and patterns into five semantic classes using DUALIST: *person*, *location*, *organization*, *date/time*, and *other* (the background or null class). The user began by inserting simple hyponym patterns (Hearst, 1992) for their corresponding classes (e.g., "people such as X" for *person*, or "organizations like X" for *organization*) and proceeded from there for 20 minutes. Since there was no gold-standard for evaluation, we randomly sampled 300 predicted extractions for each of the four non-null classes, and hired human evaluators using the Amazon Mechanical Turk service[5] to estimate precision. Each instance was assigned to three evaluators, using majority vote to score for correctness.

Table 2 shows the estimated precision, total extracted instances, and the number of user-labeled features and instances for each class. While there is room for improvement (published results for this kind of task are often above 80% precision), it is worth noting that in this experiment the user did not provide any initial "seed examples" for each class, which is fairly common in semantic class learning. In practice, such additional seeding should help, as the active learner acquired 115 labeled instances for the null class, but fewer than a dozen for each non-null class (in the first 20 minutes).

Class	Prec.	# Ext.	# Feat.	# Inst.
person	74.7	6,478	37	6
location	76.3	5,307	47	5
organization	59.7	4,613	51	7
date/time	85.7	494	51	12
other	–	32,882	13	115

Table 2: Summary of results using DUALIST for web-scale information extraction.

3.4.3 Twitter Filtering and Sentiment Analysis

There is growing interest in language analysis for online social media services such as Twitter[6] (Petrović et al., 2010; Ritter et al., 2010), which allows users to broadcast short messages limited to 140 characters. Two basic but interesting tasks in this domain are (1) language filtering and (2) sentiment classification, both of which are difficult because of the extreme brevity and informal use of language in the messages.

Even though Twitter attempts to provide language metadata for its "tweets," English is the default setting for most users, so about 35% of English-tagged tweets are actually in a different language. Furthermore, the length constraints encourage acronyms, emphatic misspellings, and orthographic shortcuts even among English-speaking users, so many tweets in English actually contain no proper English words (e.g., "OMG ur sooo gr8!! #luvya"). This may render existing lexicon-based language filters—and possibly character n-gram filters—ineffective.

To quickly build an English-language filter for Twitter, we sampled 150,000 tweets from the Twitter Streaming API and asked an annotator spend 10 minutes with DUALIST labeling English and non-English messages and features. Features were represented as unigrams and bigrams without any stopword filtering, plus a few Twitter-specific features such as emoticons (text-based representations of facial expressions such as :) or :(used to convey feeling or tone), the presence of anonymized usernames (preceded by '@') or URL links, and hashtags (compound words preceded by '#' and used to label messages, e.g., "#loveit"). Following the same methodology as Section 3.4.2, we evaluated 300 random predictions using the Mechanical Turk service. The

[5] http://www.mturk.com

[6] http://twitter.com

estimated accuracy of the trained language filter was 85.2% (inter-annotator agreement among the evaluators was 94.3%).

We then took the 97,813 tweets predicted to be in English and used them as the corpus for a sentiment classifier, which attempts to predict the mood conveyed by the author of a piece of text (Liu, 2010). Using the same feature representation as the language filter, the annotator spent 20 minutes with DUALIST, labeling tweets and features into three mood classes: *positive*, *negative*, and *neutral*. The annotator began by labeling emoticons, by which the active learner was able to uncover some interesting domain-specific salient terms, e.g., "cant wait" and "#win" for *positive* tweets or "#tiredofthat" for *negative* tweets. Using a 300-instance Mechanical Turk evaluation, the estimated accuracy of the sentiment classifier was 65.9% (inter-annotator agreement among the evaluators was 77.4%).

4 Discussion and Future Work

We have presented DUALIST, a new type of dual-strategy annotation interface for semi-supervised active learning. To support this dual-query interface, we developed a novel, fast, and practical semi-supervised learning algorithm, and demonstrated how users can employ it to rapidly develop useful natural language systems for a variety of tasks. For several of these applications, the interactively-trained systems are able to achieve 90% of state-of-the-art performance after only a few minutes of labeling effort on the part of a human annotator. By releasing DUALIST as an open-source tool, we hope to facilitate language annotation projects and encourage more user experiments in active learning.

This represents one of the first studies of an active learning system designed to compliment the strengths of both learner and annotator. Future directions along these lines include user studies of efficient annotation behaviors, which in turn might lead to new types of queries or improvements to the user interface design. An obvious extension in the natural language domain is to go beyond classification tasks and query domain knowledge for structured prediction in this way. Another interesting potential application is human-driven active feature induction and engineering, after Della Pietra et al. (1997).

From a machine learning perspective, there is an open empirical question of how useful the labels gathered by DUALIST's internal naïve Bayes model might be in later training machine learning systems with different inductive biases (e.g., MaxEnt models or decision trees), since the data are not IID. So far, attempts to "reuse" active learning data have yielded mixed results (Lewis and Catlett, 1994; Baldridge and Osborne, 2004). Practically speaking, DUALIST is designed to run on a single machine, and supports a few hundred thousand instances and features at interactive speeds on modern hardware. Distributed data storage (Chang et al., 2008) and parallelized learning algorithms (Chu et al., 2007) may help scale this approach into the millions.

Finally, modifying the learning algorithm to better cope with violated independence assumptions may be necessary for interesting language applications beyond those presented here. TAN-Trees (Friedman et al., 1997), for example, might be able to accomplish this while retaining speed and interactivity. Alternatively, one could imagine online stochastic learning algorithms for discriminatively-trained classifiers, which are semi-supervised and can exploit feature labels. To our knowledge, such flexible and efficient learning algorithms do not currently exist, but they could be easily incorporated into the DUALIST framework in the future.

Acknowledgments

Thanks to members of Carnegie Mellon's "Read the Web" research project for helpful discussions and participation in the user studies. This work is supported in part by DARPA (under contracts FA8750-08-1-0009 and AF8750-09-C-0179), the National Science Foundation (IIS-0968487), and Google.

References

J. Attenberg, P. Melville, and F. Provost. 2010. A unified approach to active dual supervision for labeling features and examples. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. Springer.

J. Baldridge and M. Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9–16. ACL Press.

J. Callan and M. Hoy. 2009. The clueweb09 dataset. http://lemurproject.org/clueweb09/.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr., and T.M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 1306–1313. AAAI Press.

F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. 2008. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems*, 26(2):1–26.

C.T. Chu, S.K. Kim, Y.A. Lin, Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun. 2007. Map-reduce for machine learning on multicore. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 281–288. MIT Press.

W. Cohen and Y. Singer. 1996. Context-sensitive learning methods for text categorization. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–315. ACM Press.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 509–516. AAAI Press.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

G. Druck, G. Mann, and A. McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602. ACM Press.

G. Druck, B. Settles, and A. McCallum. 2009. Active learning by labeling features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 81–90. ACL Press.

N. Friedman, D. Geiger, and M. Goldszmidt. 1997. Bayesian network classifiers. *Machine learning*, 29(2):131–163.

S. Godbole, A. Harpale, S. Sarawagi, and S. Chakrabarti. 2004. Document classification through interactive supervision of document and term labels. In *Proceedings of the Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 185–196. Springer.

M.A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Conference on Computational Linguistics (COLING)*, pages 539–545. ACL.

D. Heckerman. 1995. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research.

K. Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 331–339. Morgan Kaufmann.

D. Lewis and J. Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 148–156. Morgan Kaufmann.

B. Liu. 2010. Sentiment analysis and subjectivity. In N. Indurkhya and F.J. Damerau, editors, *Handbook of Natural Language Processing,*. CRC Press.

A. McCallum and K. Nigam. 1998. Employing EM in pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367. Morgan Kaufmann.

P. Melville, W. Gryc, and R.D. Lawrence. 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1275–1284. ACM Press.

S. Mohammad and T. Pedersen. 2004. Combining lexical and syntactic features for supervised word sense disambiguation. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 25–32. ACL Press.

I. Moulinier. 1996. A framework for comparing text categorization approaches. In *Proceedings of the AAAI Symposium on Machine Learning in Information Access*. AAAI Press.

A.Y. Ng and M. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 841–848. MIT Press.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up: Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86. ACL Press.

S. Petrović, M. Osborne, and V. Lavrenko. 2010. Streaming first story detection with application to Twitter. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 181–189. ACL Press.

H. Raghavan, O. Madani, and R. Jones. 2006. Active learning with feedback on both features and instances. *Journal of Machine Learning Research*, 7:1655–1686.

J.D. Rennie, L. Shih, J. Teevan, and D. Karger. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 285–295. Morgan Kaufmann.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 474–479. AAAI Press.

A. Ritter, C. Cherry, and B. Dolan. 2010. Unsupervised modeling of Twitter conversations. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 172–180. ACL Press.

T. Rose, M. Stevenson, and M. Whitehead. 2002. The Reuters corpus vol. 1 - from yesterday's news to tomorrow's language resources. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 29–31.

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

B. Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Third-order Variational Reranking on Packed-Shared Dependency Forests

Katsuhiko Hayashi[†], **Taro Watanabe**[‡], **Masayuki Asahara**[†], **Yuji Matsumoto**[†]
[†]Nara Insutitute of Science and Technology
Ikoma, Nara, 630-0192, Japan
[‡]National Institute of Information and Communications Technology
Sorakugun, Kyoto, 619-0289, Japan
{katsuhiko-h,masayu-a,matsu}@is.naist.jp
taro.watanabe@nict.go.jp

Abstract

We propose a novel *forest reranking* algorithm for discriminative dependency parsing based on a variant of Eisner's generative model. In our framework, we define two kinds of generative model for reranking. One is learned from training data offline and the other from a forest generated by a baseline parser on the fly. The final prediction in the reranking stage is performed using linear interpolation of these models and discriminative model. In order to efficiently train the model from and decode on a hypergraph data structure representing a forest, we apply extended *inside/outside* and *Viterbi* algorithms. Experimental results show that our proposed forest reranking algorithm achieves significant improvement when compared with conventional approaches.

1 Introduction

Recently, much of research on statistical parsing has been focused on k-best (or forest) reranking (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008). Typically, reranking methods first generate a list of top-k candidates (or a forest) from a baseline system, then rerank the candidates with arbitrary features that are intractable within the baseline system. In the reranking framework, the baseline system is usually modeled with a generative model, and a discriminative model is used for reranking. Sangati et al. (2009) reversed the usual order of the two models for dependency parsing by employing a generative model to rescore the k-best candidates provided by a discriminative model. They use a variant of Eisner's generative model C (Eisner, 1996b; Eisner, 1996a) for reranking and extend it to capture higher-order information than Eisner's second-order generative model. Their reranking model showed large improvements in dependency parsing accuracy. They reported that the discriminative model is very effective at filtering out bad candidates, while the generative model is able to further refine the selection among the few best candidates.

In this paper, we propose a *forest generative reranking* algorithm, opposed to Sangati et al. (2009)'s approach which reranks only k-best candidates. Forests usually encode better candidates more compactly than k-best lists (Huang, 2008). Moreover, our reranking uses not only a generative model obtained from training data, but also a sentence specific generative model learned from a forest. In the reranking stage, we use linearly combined model of these models. We call this *variational reranking* model. The model proposed in this paper is factored in the third-order structure, therefore, its non-locality makes it difficult to perform the reranking with an usual 1-best *Viterbi* search. To solve this problem, we also propose a new search algorithm, which is inspired by the third-order dynamic programming parsing algorithm (Koo and Collins, 2010). This algorithm enables us an exact 1-best reranking without any approximation. We summarize our contributions in this paper as follows.

- To extend k-best to forest generative reranking.
- We introduce *variational reranking* which is a combination approach of generative reranking and variational decoding (Li et al., 2009).
- To obtain 1-best tree in the reranking stage, we

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1479–1488,
Edinburgh, Scotland, UK, July 27–31, 2011.

propose an exact 1-best search algorithm with the third-order model.

In experiments on English Penn Treebank data, we show that our proposed methods bring significant improvement to dependency parsing. Moreover, our variational reranking framework achieves consistent improvement, compared to conventional approaches, such as simple k-best and forest-based generative reranking algorithms.

2 Dependency Parsing

Given an input sentence $x \in \mathcal{X}$, the task of statistical dependency parsing is to predict output dependencies $\hat{y}$ for x. The task is usually modeled within a discriminative framework, defined by the following equation:

$$\begin{aligned} \hat{y} &= \operatorname*{argmax}_{y \in \mathcal{Y}} s(x, y) \\ &= \operatorname*{argmax}_{y \in \mathcal{Y}} \boldsymbol{\lambda}^{\top} \cdot \mathbf{F}(y, x) \end{aligned} \quad (1)$$

where $\mathcal{Y}$ is the output space, $\boldsymbol{\lambda}$ is a parameter vector, and $\mathbf{F}()$ is a set of feature functions.

We denote a set of candidates as $G(x)$. By using $G(x)$, the conditional probability $p(y|x)$ is typically derived as follows:

$$p(y|x) = \frac{e^{\gamma \cdot s(x,y)}}{Z(x)} = \frac{e^{\gamma \cdot s(x,y)}}{\sum_{y \in G(x)} e^{\gamma \cdot s(x,y)}} \quad (2)$$

where $s(x, y)$ is the score function shown in Eq.1 and γ is a scaling factor to adjust the sharpness of the distribution and $Z(x)$ is a normarization factor.

2.1 Hypergraph Representation

We propose to encode many hypotheses in a compact representation called dependency forest. While there may be exponentially many dependency trees, the forest represents them in polynomial space. A dependency forest (or tree) can be defined as a hypergraph data strucure $\mathbf{HG}$ (Tu et al., 2010).

Figure 1 shows an example of a hypergraph for a dependency tree. A shaded hyperedge e is defined as the following form:

$$e : \langle (\mathrm{I}_{1,2}, \mathrm{girl}_{3,5}, \mathrm{with}_{5,8}), \mathrm{saw}_{1,8} \rangle.$$

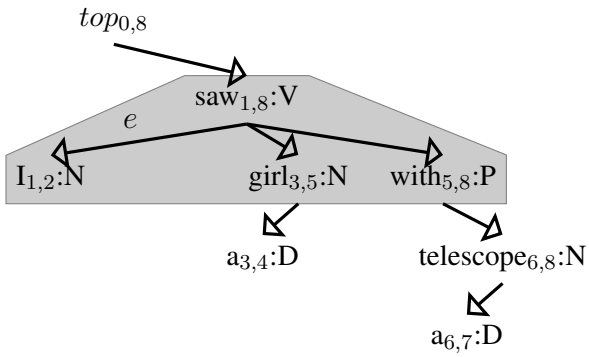


Figure 1: An example of dependency tree for a sentence "I saw a girl with a telescope".

The node saw$_{1,8}$ is a head node of e. The nodes, I$_{1,2}$, girl$_{3,5}$ and with$_{5,8}$, are tail nodes of e. The hyperedge e is an incoming edge for saw$_{1,8}$ and outgoing edge for each of I$_{1,2}$, girl$_{3,5}$ and with$_{5,8}$[1].

More formally, $\mathbf{HG}(x)$ of a forest is a pair $\langle V, E \rangle$, where V is a set of nodes and E is a set of hyperedges. Given a length m sentence $x = (w_1 \dots w_m)$, each node $v \in V$ is in the form of $w_{i,j}$ $(= (w_i \dots w_{j+1}))$ which denotes that a word w dominates the substring from positions i to j. In our implementation, each word is paired with POS-tag $tag(w)$. We denote the root node of dependency tree y as top. Each hyperedge $e \in E$ is a pair $\langle tails(e), head(e) \rangle$, where $head(e) \in V$ is the head and $tails(e) \in V^+$ are its dependants. For notational brevity of algorithmic description, we do not distinguish left and right tails in the definition, but, our implementation implicitly distinguishes left tails $tails_L(e)$ and right tails $tails_R(e)$. We define the set of incoming edges of a node v as $IE(v)$ and the set of outgoing edges of a node v as $OE(v)$.

3 Forest Reranking

3.1 Generative Model for Reranking

Given a node v in a dependency tree y, the left and right children are generated as two separate Markov sequences, each conditioned on ancestral and sibling information (context). Like a variation of Eisner's generative model C (Eisner, 1996b; Eisner, 1996a),

[1] In Figure 1, according to custom of dependency tree description, the direction of hyperedge is written as from head to tail nodes. However, in this paper, "incoming" and "outgoing" have the same meanings as those in (Huang, 2006).

Table 1: An event list of tri-sibling model whose event space is $v|h, sib, tsib, dir$, extracted from hyperedge e in Figure 1. EOC is an end symbol of sequence.

event space
I | saw NONE NONE L
EOC | saw I NONE L
girl | saw NONE NONE R
with | saw girl NONE R
EOC | saw with girl R

the probability of our model q is defined as follows:

$$q(v) = \prod_{l=1}^{|tails_L(e)|} q(v_l|\mathrm{C}(v_l)) \cdot q(v_l) \times \prod_{r=1}^{|tails_R(e)|} q(v_r|\mathrm{C}(v_r)) \cdot q(v_r) \quad (3)$$

where $|tails_L(e)|$ and $|tails_R(e)|$ are the number of left and right children of v, v_l and v_r are the left and right child of position l and r in each side. $\mathrm{C}(v)$ is a context event space of v. We explain the context event space later in more detail. The probability of the entire dependency tree y is recursively computed by $q(y(top))$ where $y(top)$ denotes a top node of y.

The probability $q(v|\mathrm{C}(v))$ is dependent on a context space $\mathrm{C}(v)$ for a node v. We define two kinds of context spaces. First, we define a tri-sibling model whose context space consists of the head node, sibling node, tri-sibling node and direction of a node v:

$$q_1(v|\mathrm{C}(v)) = q_1(v|h, sib, tsib, dir) \quad (4)$$

where h, sib and $tsib$ are head, sibling and tri-sibling node of v, and dir is a direction of v from h. Table 1 shows an example of an event list of the tri-sibling model, which is extracted from hyperedge e in Figure 1. EOC indicates the end of the left or right child sequence. This is factored in a tri-sibling structure shown in the left side of Figure 2.

Eq.4 is further decomposed into a product of the form consisting of three terms:

$$\begin{aligned} & q_1(v|h, sib, tsib, dir) \quad (5) \\ & = q_1(dist(v,h), wrd(v), tag(v)|h, sib, tsib, dir) \\ & = q_1(tag(v)|h, sib, tsib, dir) \\ & \times q_1(wrd(v)|tag(v), h, sib, tsib, dir) \\ & \times q_1(dist(v,h)|wrd(v), tag(v), h, sib, tsib, dir) \end{aligned}$$

where $tag(v)$ and $wrd(v)$ are the POS-tag and word of v and $dist(v,h)$ is the distance between positions of v and h. The values of $dist(v,h)$ are partitioned into 4 categories: $1, 2, 3-6, 7-\infty$.

Second, following Sangati et al. (2009), we define a grandsibling model whose context space consists of the head node, sibling node, grandparent node and direction of a node v.

$$q_2(v|\mathrm{C}(v)) = q_2(v|h, sib, g, dir) \quad (6)$$

where g is a grandparent node of v. Analogous to Eq.5, Eq.6 is decomposed into three terms:

$$\begin{aligned} & q_2(v|h, sib, g, dir) \quad (7) \\ & = q_2(dist(v,h), wrd(v), tag(v)|h, sib, g, dir) \\ & = q_2(tag(v)|h, sib, g, dir) \\ & \times q_2(wrd(v)|tag(v), h, sib, g, dir) \\ & \times q_2(dist(v,h)|wrd(v), tag(v), h, sib, g, dir) \end{aligned}$$

where notations are the same as those in Eq.5 with the exception of tri-sibling $tsib$ and grandparent g. This model is factored in a grandsibling structure shown in the right side of Figure 2.

The direct estimation of tri-sibling and grandsibling models from a corpus suffers from serious data sparseness issues. To overcome this, Eisner (1996a) proposed a back-off strategy which reduces the conditioning of a model. We show the reductions list for each term of two models in Table 2. The usage of reductions list is identical to Eisner (1996a) and readers may refer to it for further details.

The final prediction is performed using a log-linear interpolated model. It interpolates the baseline discriminative model and two (tri-sibling and grandsibling) generative models.

$$\hat{y} = \operatorname*{argmax}_{y \in G(x)} \sum_{n=1}^{2} \log q_n(top(y))^{\theta_n} + \log p(y|x)^{\theta_{base}} \quad (8)$$

where $\boldsymbol{\theta}$ are parameters to adjust the weight of each term in prediction. These parameters are tuned using *MERT* algorithm (Och, 2003) on development data using a criterion of accuracy maximization. The reason why we chose *MERT* is that it effectively tunes dense parameters with a line search algorithm.

Table 2: Reduction lists for tri-sibling and grandsibling models: $wt()$, $w()$ and $t()$ mean word and POS-tag, word, POS-tag for a node. d indicates the direction. The first reduction on the list keeps all or most of the original condition; later reductions throw away more and more of this information.

tri-sibling			grandsibling		
1-st term	2-nd term	3-rd term	1-st term	2-nd term	3-rd term
$wt(h)$,$wt(sib)$,$wt(tsib)$,d	$wt(h)$,$t(sib)$,d	$wt(v)$,$t(h)$,$t(sib)$,d	$wt(h)$,$wt(sib)$,$wt(g)$,d	$wt(h)$,$t(sib)$,d	$wt(v)$,$t(h)$,$t(sib)$,d
$wt(h)$,$wt(sib)$,$t(tsib)$,d	$t(h)$,$t(sib)$,d	$t(v)$,$t(h)$,$t(sib)$,d	$wt(h)$,$wt(sib)$,$t(g)$,d	$t(h)$,$t(sib)$,d	$t(v)$,$t(h)$,$t(sib)$,d
$t(h)$,$wt(sib)$,$t(tsib)$,d $wt(h)$,$t(sib)$,$t(tsib)$,d	—	—	$t(h)$,$wt(sib)$,$t(g)$,d $wt(h)$,$t(sib)$,$t(g)$,d	—	—
$t(h)$,$t(sib)$,$t(tsib)$,d	—	—	$t(h)$,$t(sib)$,$t(g)$,d	—	—

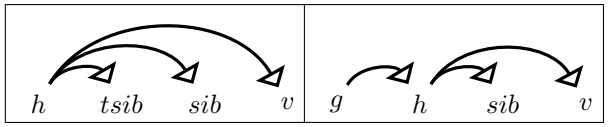


Figure 2: The left side denotes tri-sibling structure and the right side denotes grandsibling structure.

Table 3: A summarization of the model factorization and order

first-order	McDonald et al. (2005)
second-order (sibling)	Eisner (1996a) McDonald et al. (2005)
third-order (tri-sibling)	tri-sibling model Model 2 (Koo and Collins, 2010)
third-order (grandsibling)	grandsibling model (Sangati et al., 2009) Model 1 (Koo and Collins, 2010)

3.2 Exact Search Algorithm

Our baseline discriminative model uses first- and second-order features provided in (McDonald et al., 2005; McDonald and Pereira, 2006). Therefore, both our tri-sibling model and baseline discriminative model integrate local features that are factored in one hyperedge. On the other hand, the grandsibling model has non-local features because the grandparent is not factored in one hyperedge. We summarize the order of each model in Table 3. Our reranking models are generative versions of Koo and Collins (2010)'s third-order factorization model.

Non-locality of weight function makes it difficult to perform the search of Eq.8 with an usual exact *Viterbi* 1-best algorithm. One solution to resolve the intractability is an approximate k-best *Viterbi* search. For a constituent parser, Huang (2008) applied *cube pruning* techniques to forest reranking with non-local features. Cube pruning is originally proposed for the decoding of *statistical machine translation* (SMT) with an integrated n-gram language model (Chiang, 2007). It is an approximate k-best *Viterbi* search algorithm using beam search and lazy computation (Huang and Chiang, 2005).

In the case of a dependency parser, Koo and Collins (2010) proposed dynamic-programming-based third-order parsing algorithm, which enumerates all grandparents with an additional loop. Our hypergraph based search algorithm for Eq.8 share the same spirit to their third-order parsing algorithm since the grandsibling model is similar to their model 1 in that it is factored in grandsibling structure. Algorithm 1 shows the search algorithm. This is almost the same bottom-up 1-best *Viterbi* algorithm except an additional loop in line 4. Line 4 references outgoing edge e' of node h from a set of outgoing edges $OE(h)$. $tails(e)$ contains a node v, the sibling node sib and tri-sibling node $tsib$ of v, moreover, the head of e' ($head(e')$) is the grandparent for v and sib. Thus, in line 5, we can capture tri-sibling and grandsibling information and compute the current inside estimate of Eq.8.

In our actual implementation, each score of components in Eq.8 is represented as a cost. This is written as a shortest path search algorithm with a tropical (real) semiring framework (Mohri, 2002; Huang, 2006). Therefore, $\oplus$ denotes the $\min$ operater and $\otimes$ denotes the $+$ operater. The function f is defined as follows:

$$f(d(v_1, e), \ldots, d(v_{|e|}, e))) = \bigotimes_{i=1}^{|e|} d(v_i, e) \qquad (9)$$

where $d(v_i, e)$ denotes the current estimate of the best cost for a pair of node v_i and a hyperedge e. $\bigotimes$ sums the best cost of a pair of a sub span node and hyperedge e. Each c_{tsib} and c_{gsib} in line 5 and 7 indicates the cost of tri-sibling and grandsibling

Algorithm 1 Exact DP-Search Algorithm($\mathbf{HG}(x)$)

1: **for** $h \in V$ in bottom-up topological order **do**
2: **for** $e \in IE(h)$ **do**
3: // $tails(e)$ is $\{v_1, \ldots, v_{|e|}\}$.
4: **for** $e' \in OE(h)$ **do**
5: $d(h, e') = \oplus f(d(v_1, e), \ldots, d(v_{|e|}, e)) \otimes w_e \otimes c_{tsib}(h, tails(e)) \otimes c_{gsib}(head(e'), h, tails(e))$
6: **if** $h == top$ **then**
7: $d(h) = \oplus f(d(v_1, e), \ldots, d(v_{|e|}, e)) \otimes w_e \otimes c_{tsib}(h, tails(e))$

model. w_e indicates the cost of hyperedge e computed from a baseline discriminative model. Lines 6-7 denote the calculation of the best cost for a top node. We do not compute the cost of the grandsibling model when h is top node because top node has no outgoing edges.

Our baseline k-best second-order parser is implemented using Huang and Chiang (2005)'s algorithm 2 whose time complexity is $O(m^3 + mk \log k)$. Koo and Collins (2010)'s third-order parser has $O(m^4)$ time complexity and is theoretically slower than our baseline k-best parser for a long sentence. Our search algorithm is based on the third-order parsing algorithm, but, the search space is previously shrank by a baseline parser's k-best approximation and a forest pruning algorithm presented in the next section. Therefore, the time efficiency of our reranking is unimpaired.

3.3 Forest Pruning

Charniak and Johnson (2005) and Huang (2008) proposed forest pruning algorithms to reduce the size of a forest. Huang (2008)'s pruning algorithm uses a 1-best *Viterbi inside/outside* algorithm to compute an inside probability $\beta(v)$ and an outside probability $\alpha(v)$, while Charniak and Johnson (2005) use the usual *inside/outside* algorithm.

In our experiments, we use Charniak and Johnson (2005)'s forest pruning criterion because the variational model needs traditional *inside/outside* probabilities for its ML estimation. We prune away all hyperedges that have $score < \rho$ for a threshold ρ.

$$score = \frac{\alpha\beta(e)}{\beta(top)}. \tag{10}$$

Following Huang (2008), we also prune away nodes with all incoming and outgoing hyperedges pruned.

4 Variational Reranking Model

In place of a *maximum a posteriori* (MAP) decision based on Eq.2, the *minimum Bayes risk* (MBR) decision rule (Titov and Henderson, 2006) is commonly used and defined as following equation:

$$\hat{y} = \operatorname*{argmin}_{y \in G(x)} \sum_{y' \in G(x)} loss(y, y') p(y'|x) \tag{11}$$

where $loss(y, y')$ represents a loss function[2]. As an alternative to the MBR decision rule, Li et al. (2009) proposed a variational decision rule that rescores candidates with an approximate distribution $q^* \in \mathcal{Q}$.

$$\hat{y} = \operatorname*{argmax}_{y \in G(x)} q^*(y) \tag{12}$$

where q^* minimizes the KL divergence $\mathrm{KL}(p||q)$

$$\begin{aligned} q^* &= \operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{KL}(p||q) \\ &= \operatorname*{argmax}_{q \in \mathcal{Q}} \sum_{y \in G(x)} p \log q \end{aligned} \tag{13}$$

where each p and q represents $p(y|x)$ and $q(y)$. For SMT systems, q^* is modeled by n-gram language model over output strings. While the decoding based on q^* is an approximation of intractable MAP decoding[3], it works as a rescoring function for candidates generated from a baseline model. Here, we propose to apply the variational decision rule to dependency parsing. For dependency parsing, we can choose to model q^* as the tri-sibling and grandsibling generative models in section 3.

[2] In case of dependency parsing, Titov and Henderson (2006) proposed that a loss function is simply defined using a dependency attachment score.

[3] In SMT, a marginalization of all derivations which yield a paticular translation needs to be carried out for each translation. This makes the MAP decoding NP-hard in SMT. This variational approximate framework can be applied to other tasks collapsing *spurious ambiguity*, such as latent-variable parsing (Matsuzaki et al., 2005).

Algorithm 2 DP-ML Estimation($\mathbf{HG}(x)$)

```
1: run inside and outside algorithm on HG(x)
2: for v ∈ V do
3:   for e ∈ IE(v) do
4:     c_tsib = p_e · α(v)/β(top)
5:     for u ∈ tails(e) do
6:       c_tsib = c_tsib · β(u)
7:       for e' ∈ IE(u) do
8:         c_gsib = p_e · p_e' · α(v)/β(top)
9:         for u' ∈ tails(e) \ u do
10:          c_gsib = c_gsib · β(u')
11:        for u'' ∈ tails(e') do
12:          c_gsib = c_gsib · β(u'')
13:        for u'' ∈ tails(e') do
14:          c̄2(u''|C(u''))+ = c_gsib
15:          c̄2(C(u''))+ = c_gsib
16:    for u ∈ tails(e) do
17:      c̄1(u|C(u))+ = c_tsib
18:      c̄1(C(u))+ = c_tsib
19: MLE estimate q1*, q2* using formula Eq.14
```

4.1 ML Estimation from a Forest

$q^*(v|\mathrm{C}(v))$ is estimated from a forest using a *maximum likelihood estimation* (MLE). The count of events is no longer an integer count, but an expected count under p, which is formulated as follows:

$$q^*(v|\mathrm{C}(v)) = \frac{\overline{c}(v|\mathrm{C}(v))}{\overline{c}(\mathrm{C}(v))} = \frac{\sum_y p(y|x) c_{v|\mathrm{C}(v)}(y)}{\sum_y p(y|x) c_{\mathrm{C}(v)}(y)} \quad (14)$$

where $c_{\mathrm{e}}(y)$ is the number of event e in y. The estimation of Eq.14 can be efficiently performed on a hypergraph data structure $\mathbf{HG}(x)$ of a forest.

Algorithm 2 shows the estimation algorithm. First, it runs the *inside/outside* algorithm on $\mathbf{HG}(x)$. We denote inside weight for a node v as $\beta(v)$ and outside weight as $\alpha(v)$. For each hyperedge e, we denote c_{tsib} as the posterior weight for computing expected count $\overline{c}_1$ of events in the tri-sibling model q_1^*. Lines 16-18 compute $\overline{c}_1$ for all events occuring in a hyperedge e.

The expected count $\overline{c}_2$ needed for the estimation of grandsibling model q_2^* is extracted in lines 7-15. $\overline{c}_2$ for a grandsibling model must be extracted over two hyperedges e and e' because it needs grandparent information. Lines 8-12 show the algorithm to compute the posterior weight c_{gsib} of e and e', which is similar to that to compute the posterior weight of rules of *tree substitution grammars* used in tree-based MT systems (Mi and Huang, 2008). Lines 13-15 compute expected counts $\overline{c}_2$ of events occuring over two hyperedges e and e'. Finally, line 19 estimates q_1^* and q_2^* using the form in Eq.14.

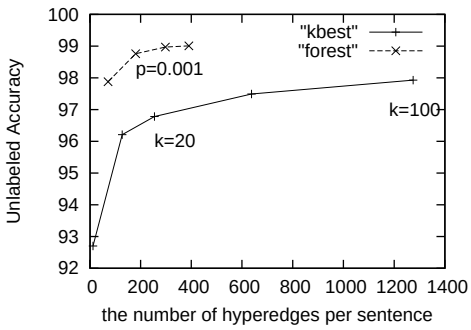


Figure 3: The relationship between tha data size (the number of hyperedges) and oracle scores on development data: Forests encode candidates with high accuracy scores more compactly than k-best lists.

Li et al. (2009) assumes n-gram locality of the forest to efficiently train the model, namely, the baseline n-gram model has larger n than that of variational n-gram model. In our case, grandsibling locality is not embedded in the forest generated from the baseline parser. Therefore, we need to reference incoming hyperedges of tail nodes in line 7.

y^* of Eq.12 may be locally appropriate but globally inadequate because q^* only approximates p. Therefore, we log-linearly combine q^* with a global generative model estimated from the training data and the baseline discriminative model.

$$\hat{y} = \operatorname*{argmax}_{y \in G(x)} \sum_{n=1}^{2} \log q_n(top(y))^{\theta_n} + \sum_{n=1}^{2} \log q_n^*(top(y))^{\theta_n^*} + \log p(y|x)^{\theta_{base}} \quad (15)$$

Algorithm1 is also applicable to the decoding of Eq.15. Note that this framework is a combination of variational decoding and generative reranking. We call this framework *variational reranking*.

Table 4: The statistics of forests and 20-best lists on development data: this shows the average number of hyperedges and nodes per sentence and oracle scores.

	forest	20-best
pruning threshold	$\rho = 10^{-3}$	—
ave. num of hyperedges	180.67	255.04
ave. num of nodes	135.74	491.42
oracle scores	98.76	96.78

5 Experiments

Experiments are performed on English Penn Treebank data. We split WSJ part of the Treebank into sections 02-21 for training, sections 22 for development, sections 23 for testing. We use Yamada and Matsumoto (2003)'s head rules to convert phrase structure to dependency structure. We obtain k-best lists and forests generated from the baseline discriminative model which has the same feature set as provided in (McDonald et al., 2005), using the second-order Eisner algorithms. We use MIRA for training as it is one of the learning algorithms that achieves the best performance in dependency parsing. We set the scaling factor $\gamma = 1.0$.

We also train a generative reranking model from the training data. To reduce the data sparseness problem, we use the back-off strategy proposed in (Eisner, 1996a). Parameters $\boldsymbol{\theta}$ are trained using MERT (Och, 2003) and for each sentence in the development data, 300-best dependency trees are extracted from its forest. Our variational reranking does not need much time to train the model because the training is performed over not the training data (39832 sentences) but the development data (1700 sentences)[4]. After MERT was performed until the convergence, the variational reranking finally achieved a 94.5 accuracy score on development data.

5.1 k-best Lists vs. Forests

Figure 3 shows the relationship between the size of data structure (the number of hyperedges) and accuracy scores on development data. Obviously, forests can encode a large number of potential candidates more compactly than k-best lists. This means that for reranking, there is more possibility of selecting good candidates in forests than k-best lists.

Table 4 shows the statistics of forests and 20-best lists on development data. This setting, threshold $\rho = 10^{-3}$ for pruning, is also used for testing. Forests, which have an average of 180.67 hyperedges per sentence, achieve oracle score of 98.76, which is about 1.0% higher than the 96.78 oracle score of 20-best lists with 255.04 hyperedges per sentence. Though the size of forests is smaller than that of k-best lists, the oracle scores of forests are much higher than those of k-best lists.

5.2 The Performance of Reranking

First, we compare the performance of variational decoding with that of MBR decoding. The results are shown in Table 5. Variational decoding outperforms MBR decodings. However, compared with baseline, the gains of variational and MBR decoding are small. Second, we also compare the performance of variational reranking with k-best and forest generative reranking algorithms. Table 6 shows that our variational reranking framework achieves the highest accuracy scores.

Being different from the decoding framework, reranking achieves significant improvements. This result is intuitively reasonable because the reranking model obtained from training data has the ability to select a globally consistent candidate, while the variational approximate model obtained from a forest only supports selecting a locally consistent candidate. On the other hand, the fact that variational reranking achieves the best results clearly indicates that the combination of sentence specific generative model and that obtained from training data is successful in selecting both locally and globally appropriate candidate from a forest.

Table 7 shows the parsing time (on 2.66GHz Quad-Core Xeon) of the baseline k-best, generative reranking and variational reranking parsers (java implemented). The variational reranking parser contains the following procedures.

1. k-best forest creation (baseline)
2. Estimation of variational model
3. Forest pruning
4. Search with the third-order model

Our reranking parser incurred little overhead to the

[4]To generate forests, sentences are parsed only once before the training. MERT is performed over the forests. We can also apply a more efficient hypergraph MERT algorithm (Kumar et al., 2009) to the training than a simple MERT algorithm.

Table 5: The comparison of the decoding frameworks: MBR decoding seeks a candidate which has the highest accuracy scores over a forest (Kumar et al., 2009). Variational decoding is performed based on Eq.8.

Decoding \ Eval	Unlabeled
baseline	91.9
MBR (8-best forest)	91.99
Variational (8-best forest)	**92.17**

Table 6: The comparison of the reranking frameworks: Generative means k-best or forest reranking algorithm based on a generative model estimated from a corpus. Variational reranking is performed based on Eq.15.

Reranking \ Eval	Unlabeled
Generative (8-best)	92.66
Generative (8-best forest)	92.72
Variational (8-best forest)	**92.87**

Table 7: The parsing time (CPU second per sentence) and accuracy score of the baseline k-best, generative reranking and variational reranking parsers

k	baseline	generative	variational
2	0.09 (91.9)	+0.03 (92.67)	+0.05 (92.76)
4	0.1 (91.9)	+0.05 (92.68)	+0.09 (92.81)
8	0.13 (91.9)	+0.06 (92.72)	+0.11 (92.87)
16	0.18 (91.9)	+0.07 (92.75)	+0.12 (92.89)
32	0.29 (91.9)	+0.07 (92.73)	+0.13 (92.89)
64	0.54 (91.9)	+0.08 (92.72)	+0.15 (92.87)

Table 8: The comparison of tri-sibling and grandsibling models: the performance of the grandsibling model outperforms that of the tri-sibling model.

Model \ Eval	Unlabeled
tri-sibling	92.63
grandsibling	92.74

Table 10: Comparison of our best result (using 16-best forests) with other best-performing Systems on the whole section 23

Parser	English
McDonald et al. (2005)	90.9
McDonald and Pereira (2006)	91.5
Koo et al. (2008) standard	92.02
Huang and Sagae (2010)	92.1
Koo and Collins (2010) model1	93.04
Koo and Collins (2010) model2	92.93
this work	92.89
Koo et al. (2008) semi-sup	93.16
Suzuki et al. (2009)	93.79

baseline parser in terms of runtime. This means that our reranking parser can parse sentences at reasonable times.

5.3 The Effects of Third-order Factors and Error Analysis

From results in section 5.2, our variational reranking model achieves higher accuracy scores than the others. To analyze the factors that improve accuracy scores, we further investigate whether variational reranking is performed better with the tri-sibling or grandsibling model. Table 8 indicates that grandsibling model achieves a larger gain than that of tri-sibling model. Table 9 shows the examples whose accuracy scores improved by the grandsibling model. For example, the dependency relationship from *Verb* to *Noun phrase* was corrected by our proposed model.

On the other hand, many errors remain still in our results. In our experiments, 48% of sentences which contain errors have *Prepositional* word errors. In fact, well-known *PP-Attachment* is a problem to be solved for natural language parsers. Other remaining errors are caused by symbols such as .,:“”(). 45% sentences contain such a dependency mistake. Adding features to solve these problems may potentially improve our parser more.

5.4 Comparison with Other Systems

Table 10 shows the comparison of the performance of variational reranking (16-best forests) with that of other systems. Our method outperforms supervised parsers with second-order features, and achieves comparable results compared to a parser with third-order features (Koo and Collins, 2010). We can not directly compare our method with semi-supervised parsers such as Koo et al. (2008)'s semi-sup and Suzuki et al. (2009), because ours does not use additional unlabeled data for training. The model trained from unlabeled data can be easily incorporated into our reranking framework. We plan to investigate semi-supervised learning in future work.

Table 9: Examples of outputs for input sentence No.148 and No.283 in section 23 from baseline and variational reranking parsers. The underlined portions show the effect of the grandsibling model.

sent (No.148)	A quick turnaround is crucial to Quantum because its cash requirements remain heavy .													
correct	3	3	4	0	4	5	6	4	11	11	12	8	12	4
baseline	3	3	4	0	4	5	6	4	11	11	8	8	12	4
proposed	3	3	4	0	4	5	6	4	11	11	12	8	12	4
sent (No.283)	Many called it simply a contrast in styles .													
correct	2	0	2	6	6	2	6	7	2					
baseline	2	0	2	2	6	2	6	7	2					
proposed	2	0	2	6	6	2	6	7	2					

6 Related Work

Collins (2000) and Charniak and Johnson (2005) proposed a reranking algorithm for constituent parsers. Huang (2008) extended it to a forest reranking algorithm with non-local features. Our framework is for a dependency parser and the decoding in the reranking stage is done with an exact 1-best dynamic programming algorithm. Sangati et al. (2009) proposed a k-best generative reranking algorithm for dependency parsing. In this paper, we use a similar generative model, but combined with a variational model learned on the fly. Moreover, our framework is applicable to forests, not k-best lists.

Koo and Collins (2010) presented third-order dependency parsing algorithm. Their model 1 is defined by an enclosing grandsibling for each sibling or grandchild part used in Carreras (2007). Our grandsibling model is similar to the model 1, but ours is defined by a generative model. The decoding in the reranking stage is also similar to the parsing algorithm of their model 1. In order to capture grandsibling factors, our decoding calculates inside probablities for not the current head node but each pair of the node and its outgoing edges.

Titov and Henderson (2006) reported that the MBR approach could be applied to a projective dependency parser. In the field of SMT, for an approximation of MAP decoding, Li et al. (2009) proposed variational decoding and Kumar et al. (2009) presented hypergraph MBR decoding. Our variational model is inspired by the study of Li et al. (2009) and we apply it to a dependency parser in order to select better candidates with third-order information. We also propose an efficient algorithm to estimate the non-local third-order model structure.

7 Conclusions

In this paper, we propose a novel forest reranking algorithm for dependency parsing. Our reranking algorithm is a combination approach of generative reranking and variational decoding. The search algorithm in the reranking stage can be performed using dynamic programming algorithm. Our variational reranking is aimed at selecting a candidate from a forest, which is correct both in local and global. Our experimental results show more significant improvements than conventional approaches, such as k-best and forest generative reranking.

In the future, we plan to investigate more appropriate generative models for reranking. *PP-Attachment* is one of the most difficult problems for a natural language parser. We plan to examine to model such a complex structure (granduncle) (Goldberg and Elhadad, 2010) or higher-order structure than third-order for reranking which is computationally expensive for a baseline parser. As we mentioned in Section 5.4, we also plan to incorporate semi-supervised learning into our framework, which may potentially improve our reranking performance.

Acknowledgments

We would like to thank Graham Neubig and Masashi Shimbo for their helpful comments and to the anonymous reviewers for their effort of reviewing our paper and giving valuable comments. This work was supported in part by Grant-in-Aid for Japan Society for the Promotion of Science (JSPS) Research Fellowship for Young Scientists.

References

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

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. the 43rd ACL*, pages 173–180.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33:201–228.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. the ICML*.

J. M. Eisner. 1996a. An empirical comparison of probability models for dependency grammar. In *Technical Report*, pages 1–18.

J. M. Eisner. 1996b. Three new probabilistic models for dependency parsing: An exploration. In *Proc. the 16th COLING*, pages 340–345.

Y. Goldberg and M. Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proc. the HLT-NAACL*, pages 742–750.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proc. the IWPT*, pages 53–64.

L. Huang and K. Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proc. the ACL*, pages 1077–1086.

L. Huang. 2006. Dynamic programming algorithms in semiring and hypergraph frameworks. *Qualification Exam Report*, pages 1–19. http://www.cis.upenn.edu/ lhuang3/wpe2/.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. the 46th ACL*, pages 586–594.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proc. the 48th ACL*, pages 1–11.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. the ACL*, pages 595–603.

S. Kumar, W. Macherey, C. Dyer, and F. Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proc. the 47th ACL*, pages 163–171.

Z. Li, J. Eisner, and S. Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proc. the 47th ACL*, pages 593–601.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proc. the ACL*, pages 75–82.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. EACL*, pages 81–88.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. the 43rd ACL*, pages 91–98.

H. Mi and L. Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*, pages 206–214.

M. Mohri. 2002. Semiring framework and algorithms for shortest-distance problems. *Automata, Languages and Combinatorics*, 7:321–350.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. the 41st ACL*, pages 160–167.

F. Sangati, W. Zuidema, and R. Bod. 2009. A generative re-ranking model for dependency parsing. In *Proc. the 11th IWPT*, pages 238–241.

J. Suzuki, H. Isozaki, X. Carreras, and M. Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proc. the EMNLP*, pages 551–560.

I. Titov and J. Henderson. 2006. Bayes risk minimization in natural language parsing. In *Technical Report*, pages 1–9.

Z. Tu, Y. Liu, Y. Hwang, Q. Liu, and S. Lin. 2010. Dependency forest for statistical machine translation. In *Proc. the 23rd COLING*, pages 1092–1100.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. the IWPT*, pages 195–206.

Training dependency parsers by jointly optimizing multiple objectives

Keith Hall Ryan McDonald Jason Katz-Brown Michael Ringgaard
Google Research
{kbhall|ryanmcd|jasonkb|ringgaard}@google.com

Abstract

We present an online learning algorithm for training parsers which allows for the inclusion of multiple objective functions. The primary example is the extension of a standard supervised parsing objective function with additional loss-functions, either based on intrinsic parsing quality or task-specific extrinsic measures of quality. Our empirical results show how this approach performs for two dependency parsing algorithms (graph-based and transition-based parsing) and how it achieves increased performance on multiple target tasks including reordering for machine translation and parser adaptation.

1 Introduction

The accuracy and speed of state-of-the-art dependency parsers has motivated a resumed interest in utilizing the output of parsing as an input to many downstream natural language processing tasks. This includes work on question answering (Wang et al., 2007), sentiment analysis (Nakagawa et al., 2010), MT reordering (Xu et al., 2009), and many other tasks. In most cases, the accuracy of parsers degrades when run on out-of-domain data (Gildea, 2001; McClosky et al., 2006; Blitzer et al., 2006; Petrov et al., 2010). But these accuracies are measured with respect to gold-standard out-of-domain parse trees. There are few tasks that actually depend on the complete parse tree. Furthermore, when evaluated on a downstream task, often the optimal parse output has a model score lower than the best parse as predicted by the parsing model. While this means that we are not properly modeling the downstream task in the parsers, it also means that there is some information from small task or domain-specific data sets which could help direct our search for optimal parameters during parser training. The goal being not necessarily to obtain better parse performance, but to exploit the structure induced from human labeled treebank data while targeting specific extrinsic metrics of quality, which can include task specific metrics or external weak constraints on the parse structure.

One obvious approach to this problem is to employ parser reranking (Collins, 2000). In such a setting, an auxiliary reranker is added in a pipeline following the parser. The standard setting involves training the base parser and applying it to a development set (this is often done in a cross-validated jack-knife training framework). The reranker can then be trained to optimize for the downstream or extrinsic objective. While this will bias the reranker towards the target task, it is limited by the oracle performance of the original base parser.

In this paper, we propose a training algorithm for statistical dependency parsers (Kübler et al., 2009) in which a single model is jointly optimized for a regular supervised training objective over the treebank data as well as a task-specific objective – or more generally an extrinsic objective – on an additional data set. The case where there are both gold-standard trees and a task-specific objective for the entire training set is a specific instance of the larger problem that we address here. Specifically, the algorithm takes the form of an online learner where a training instance is selected and the param-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1489–1499,
Edinburgh, Scotland, UK, July 27–31, 2011.

eters are optimized based on the objective function associated with the instance (either intrinsic or extrinsic), thus jointly optimizing multiple objectives. An update schedule trades-off the relative importance of each objective function. We call our algorithm *augmented-loss training* as it optimizes multiple losses to augment the traditional supervised parser loss.

There have been a number of efforts to exploit weak or external signals of quality to train better prediction models. This includes work on generalized expectation (Mann and McCallum, 2010), posterior regularization (Ganchev et al., 2010) and constraint driven learning (Chang et al., 2007; Chang et al., 2010). The work of Chang et al. (2007) on constraint driven learning is perhaps the closest to our framework and we draw connections to it in Section 5. In these studies the typical goal is to use the weak signal to improve the structured prediction models on the intrinsic evaluation metrics. For our setting this would mean using weak application specific signals to improve dependency parsing. Though we explore such ideas in our experiments, in particular for semi-supervised domain adaptation, we are primarily interested in the case where the weak signal is precisely what we wish to optimize, but also desire the benefit from using both data with annotated parse structures and data specific to the task at hand to guide parser training.

In Section 2 we outline the augmented-loss algorithm and provide a convergence analysis. In Section 3 and 4 we present a set of experiments defining diffent augmented losses covering a task-specific extrinsic loss (MT reordering), a domain adaptation loss, and an alternate intrinsic parser loss. In all cases we show the augmented-loss framework can lead to significant gains in performance. In Section 5 we tie our augmented-loss algorithm to other frameworks for encoding auxiliary information and/or joint objective optimization.

2 Methodology

We present the augmented-loss algorithm in the context of the structured perceptron. The structured perceptron (Algorithm 1) is an on-line learning algorithm which takes as input: 1) a set of training examples $d_i = (x_i, y_i)$ consisting of an input sentence x_i and an output y_i; and 2) a loss-function, $L(\hat{y}, y)$, that measures the cost of predicting output $\hat{y}$ relative to the gold standard y and is usually the 0/1 loss (Collins, 2002). For dependency parser training, this set-up consists of input sentences x and the corresponding gold dependency tree $y \in \mathcal{Y}_x$, where $\mathcal{Y}_x$ is the space of possible parse trees for sentence x. In the perceptron setting, $F_\theta(x) = \arg\max_{y \in \mathcal{Y}_x} \theta \cdot \Phi(y)$ where Φ is mapping from a parse tree y for sentence x to a high dimensional feature space. Learning proceeds by predicting a structured output given the current model, and if that structure is incorrect, updating the model: rewarding features that fire in the gold-standard $\Phi(y_i)$, and discounting features that fire in the predicted output, $\Phi(\hat{y}_i)$.

Algorithm 1 Structured Perceptron

```
{Input data sets: D = {d_1 = (x_1, y_1) ... d_N = (x_N, y_N)}}
{Input 0/1 loss: L(F_θ(x), y) = [F_θ(x) ≠ y ? 1 : 0]}
{Let: F_θ(x) = arg max_{y∈Y} θ · Φ(y)}
{Initialize model parameters: θ = 0⃗}
repeat
  for i = 1 ... N do
    {Compute structured loss}
    ŷ_i = F_θ(x_i)
    if L(ŷ_i, y_i) > 0 then
      {Update model Parameters}
      θ = θ + Φ(y_i) − Φ(ŷ_i)
    end if
  end for
until converged
{Return model θ}
```

The structured perceptron, as given in Algorithm 1, only updates when there is a positive loss, meaning that there was a prediction mistake. For the moment we will abstract away from details such as the precise definition of $F(x)$ and $\Phi(y)$. We will show in the next section that our augmented-loss method is general and can be applied to any dependency parsing framework that can be trained by the perceptron algorithm, such as transition-based parsers (Nivre, 2008; Zhang and Clark, 2008) and graph-based parsers (McDonald et al., 2005).

2.1 Augmented-Loss Training

The augmented-loss training algorithm that we propose is based on the structured perceptron; however, the augmented-loss training framework is a general

mechanism to incorporate multiple loss functions in online learner training. Algorithm 2 is the pseudo-code for the augmented-loss structured perceptron algorithm. The algorithm is an extension to Algorithm 1 where there are 1) multiple loss functions being evaluated $L^1, \ldots, L^M$; 2) there are multiple datasets associated with each of these loss functions $D^1, \ldots, \mathcal{D}^M$; and 3) there is a schedule for processing examples from each of these datasets, where Sched(j, i) is true if the j^{th} loss function should be updated on the i^{th} iteration of training. Note that for data point $d_i^j = (x, y)$, which is the i^{th} training instance of the j^{th} data set, that y does not necessarily have to be a dependency tree. It can either be a task-specific output of interest, a partial tree, or even null, in the case where learning will be guided strictly by the loss L^j. The training algorithm is effectively the same as the perceptron, the primary difference is that if L^j is an extrinsic loss, we cannot compute the standard updates since we do not necessarily know the correct parse (the line indicated by †). Section 2.2 shows one method for updating the parser parameters for extrinsic losses.

In the experiments in this paper, we only consider the case where there are two loss functions: a supervised dependency parsing labeled-attachment loss; and an additional loss, examples of which are presented in Section 3.

Algorithm 2 Augmented-Loss Perceptron

```
{Input data sets}:
  D^1 = {d_1^1 = (x_1^1, y_1^1) ... d_{N^1}^1 = (x_{N^1}^1, y_{N^1}^1)},
  ...
  D^M = {d_1^M = (x_1^M, y_1^M) ... d_{N^M}^M = (x_{N^M}^M, y_{N^M}^M)}
{Input loss functions: L^1 ... L^M}
{Initialize indexes: c_1 ... c_M = 0⃗}
{Initialize model parameters: θ = 0⃗}
i = 0
repeat
  for j = 1 ... M do
    {Check whether to update L^j on iteration i}
    if Sched(j, i) then
      {Compute index of instance – reset if c_j ≡ N^j}
      c_j = [(c_j ≡ N^j) ? 0 : c_j + 1]
      {Compute structured loss for instance}
      if L^j is intrinsic loss then
        ŷ = F_θ(x_{c_j}^j)
        if L^j(ŷ, y_{c_j}^j) > 0 then
          θ = θ + Φ(y_{c_j}^j) − Φ(ŷ)      {y_{c_j}^j is a tree}
        end if
      else if L^j is an extrinsic loss then
        {See Section 2.2}†
      end if
    end if
  end for
  i = i + 1
until converged
{Return model θ}
```

2.2 Inline Ranker Training

In order to make Algorithm 2 more concrete, we need a way of defining the loss and resulting parameter updates for the case when L^j is not a standard supervised parsing loss († from Algorithm 2). Assume that we have a cost function $C(x_i, \hat{y}, y_i)$ which, given a training example (x_i, y_i) will give a score for a parse $\hat{y} \in \mathcal{Y}_{x_i}$ relative to some output y_i. While we can compute the score for any parse, we are unable to determine the features associated with the optimal parse, as y_i need not be a parse tree. For example, consider a machine translation reordering system which uses the parse $\hat{y}$ to reorder the words of x_i, the optimal reordering being y_i. Then $C(x_i, \hat{y}, y_i)$ is a reordering cost which is large if the predicted parse induces a poor reordering of x_i.

We propose a general purpose loss function which is based on parser k-best lists. The inline reranker uses the currently trained parser model θ to parse the external input, producing a k-best set of parses: $F_\theta^{\text{k-best}}(x_i) = \{\hat{y}_1, \ldots, \hat{y}_k\}$. We can compute the cost function $C(x_i, \hat{y}, y_i)$ for all $\hat{y} \in F_\theta^{\text{k-best}}(x_i)$. If the 1-best parse, $\hat{y}_1$, has the lowest cost, then there is no lower cost parse in this k-best list. Otherwise, the lowest-cost parse in $F_\theta^{\text{k-best}}(x_i)$ is taken to be the *correct* output structure y_i, and the 1-best parse is taken to be an incorrect prediction. We can achieve this by substituting the following into Algorithm 2 at line †.

Algorithm 3 Reranker Loss

```
{ŷ_1, ..., ŷ_k} = F_θ^{k-best}(x_i)
τ = min_τ C(x_{c_j}^j, ŷ_τ, y_{c_j}^j)    {τ is min const index}
L^j(ŷ_1, y_{c_j}^j) = C(x_{c_j}^j, ŷ_1, y_{c_j}^j) − C(x_{c_j}^j, ŷ_τ, y_{c_j}^j)
if L^j(ŷ_1, y_{c_j}^j) > 0 then
  θ = θ + Φ(ŷ_τ) − Φ(ŷ_1)
end if
```

Again the algorithm only updates when there is an error – when the 1-best output has a higher cost than any other output in the k-best list – resulting

in positive L^j. The intuition behind this method is that in the presence of only a cost function and a k-best list, the parameters will be updated towards the parse structure that has the lowest cost, which over time will move the parameters of the model to a place with low extrinsic loss.

We exploit this formulation of the general-purpose augmented-loss function as it allows one to include any extrinsic cost function which is dependent of parses. The scoring function used does not need to be factored, requiring no internal knowledge of the function itself. Furthermore, we can apply this to any parsing algorithm which can generate k-best lists. For each parse, we must retain the features associated with the parse (e.g., for transition-based parsing, the features associated with the transition sequence resulting in the parse).

There are two significant differences from the inline reranker loss function and standard reranker training. First, we are performing this decision per example as each data item is processed (this is done in the inner loop of the Algorithm 2). Second, the feedback function for selecting a parse is based on an external objective function. The second point is actually true for many minimum-error-rate training scenarios, but in those settings the model is updated as a post-processing stage (after the base-model is trained).

2.3 Convergence of Inline Ranker Training

A training set $\mathcal{D}$ is loss-separable with margin $\gamma > 0$ if there exists a vector u with $\|u\| = 1$ such that for all $y', y'' \in \mathcal{Y}_x$ and $(x, y) \in \mathcal{D}$, if $L(y', y) < L(y'', y)$, then $u \cdot \Phi(y') - u \cdot \Phi(y'') \geq \gamma$. Furthermore, let $R \geq ||\Phi(y) - \Phi(y')||$, for all y, y'.

Assumption 1. *Assume training set $\mathcal{D}$ is loss-separable with margin γ.*

Theorem 1. *Given Assumption 1. Let m be the number of mistakes made when training the perceptron (Algorithm 2) with inline ranker loss (Algorithm 3) on $\mathcal{D}$, where a mistake occurs for $(x, y) \in \mathcal{D}$ with parameter vector θ when $\exists \hat{y}_j \in F_\theta^{k\text{-best}}(x)$ where $\hat{y}_j \neq \hat{y}_1$ and $L(\hat{y}_j, y) < L(\hat{y}_1, y)$. If training is run indefinitely, then $m \leq \frac{R^2}{\gamma^2}$.*

Proof. Identical to the standard perceptron proof, e.g., Collins (2002), by inserting in loss-separability for normal separability. □

Like the original perceptron theorem, this implies that the algorithm will converge. However, unlike the original theorem, it does not imply that it will converge to a parameter vector θ such that for all $(x, y) \in \mathcal{D}$, if $\hat{y} = \arg\max_{\hat{y}} \theta \cdot \Phi(\hat{y})$ then $L(\hat{y}, y) = 0$. Even if we assume for every x there exists an output with zero loss, Theorem 1 still makes no guarantees. Consider a training set with one instance (x, y). Now, set $k = 2$ for the k-best output list and let $\hat{y}_1$, $\hat{y}_2$, and $\hat{y}_3$ be the top-3 scoring outputs and let $L(\hat{y}_1, y) = 1$, $L(\hat{y}_2, y) = 2$ and $L(\hat{y}_3, y) = 0$. In this case, no updates will ever be made and $\hat{y}_1$ will remain unchanged even though it doesn't have minimal loss. Consider the following assumption:

Assumption 2. *For any parameter vector θ that exists during training, either 1) for all $(x, y) \in \mathcal{D}$, $L(\hat{y}_1, y) = 0$ (or some optimal minimum loss), or 2) there exists at least one $(x, y) \in \mathcal{D}$ where $\exists \hat{y}_j \in F_\theta^{k\text{-best}}(x)$ such that $L(\hat{y}_j, y) < L(\hat{y}_1, y)$.*

Assumption 2 states that for any θ that exists during training, but before convergence, there is at least one example in the training data where k is large enough to include one output with a lower loss when $\hat{y}_1$ does not have the optimal minimal loss. If $k = \infty$, then this is the standard perceptron as it guarantees the optimal loss output to be in the k-best list. But we are assuming something much weaker here, i.e., not that the k-best list will include the minimal loss output, only a single output with a lower loss than the current best guess. However, it is strong enough to show the following:

Theorem 2. *Given Assumption 1 and Assumption 2. Training the perceptron (Algorithm 2) with inline ranker loss (Algorithm 3) on $\mathcal{D}$ 1) converges in finite time, and 2) produces parameters θ such that for all $(x, y) \in \mathcal{D}$, if $\hat{y} = \arg\max_{\hat{y}} \theta \cdot \Phi(\hat{y})$ then $L(\hat{y}, y) = 0$ (or equivalent minimal loss).*

Proof. It must be the case for all $(x, y) \in \mathcal{D}$ that $L(\hat{y}_1, y) = 0$ (and $\hat{y}_1$ is the argmax) after a finite amount of time. Otherwise, by Assumption 2, there exists some x, such that when it is next processed, there would exist an output in the k-best list that had a lower loss, which will result in an additional mistake. Theorem 1 guarantees that this can not continue indefinitely as the number of mistakes is bounded. □

Thus, the perceptron algorithm will converge to optimal minimal loss under the assumption that k is large enough so that the model can keep improving. Note that this does not mean k must be large enough to include a zero or minimum loss output, just large enough to include a better output than the current best hypothesis. Theorem 2, when coupled with Theorem 1, implies that augmented-loss learning will make at most R^2/γ^2 mistakes at training, but does not guarantee the rate at which these mistakes will be made, only that convergence is finite, providing that the scheduling time (defined by Sched()) between seeing the same instance is always finite, which is always true in our experiments.

This analysis does not assume anything about the loss L. Every instance (x, y) can use a different loss. It is only required that the loss for a specific input-output pair is fixed throughout training. Thus, the above analysis covers the case where some training instances use an extrinsic loss and others an intrinsic parsing loss. This also suggests more efficient training methods when extracting the k-best list is prohibitive. One can parse with $k = 2, 4, 8, 16, \ldots$ until an k is reached that includes a lower loss parse. It may be the case that for most instances a small k is required, but the algorithm is doing more work unnecessarily if k is large.

3 Experimental Set-up

3.1 Dependency Parsers

The augmented-loss framework we present is general in the sense that it can be combined with any loss function and any parser, provided the parser can be parameterized as a linear classifier, trained with the perceptron and is capable of producing a k-best list of trees. For our experiments we focus on two dependency parsers.

- **Transition-based**: An implementation of the transition-based dependency parsing framework (Nivre, 2008) using an arc-eager transition strategy and are trained using the perceptron algorithm as in Zhang and Clark (2008) with a beam size of 8. Beams with varying sizes can be used to produce k-best lists. The features used by all models are: the part-of-speech tags of the first four words on the buffer and of the top two words on the stack; the word identities of the first two words on the buffer and of the top word on the stack; the word identity of the syntactic head of the top word on the stack (if available); dependency arc label identities for the top word on the stack, the left and rightmost modifier of the top word on the stack, and the left most modifier of the first word in the buffer (if available). All feature conjunctions are included.

- **Graph-based**: An implementation of graph-based parsing algorithms with an arc-factored parameterization (McDonald et al., 2005). We use the non-projective k-best MST algorithm to generate k-best lists (Hall, 2007), where $k = 8$ for the experiments in this paper. The graph-based parser features used in the experiments in this paper are defined over a word, w_i at position i; the head of this word $w_{\rho(i)}$ where $\rho(i)$ provides the index of the head word; and part-of-speech tags of these words t_i. We use the following set of features similar to McDonald et al. (2005):

isolated features:	$w_i, t_i, w_{\rho(i)}, t_{\rho(i)}$
word-tag pairs:	$(w_i, t_i); (w_{\rho(i)}, t_{\rho(i)})$
word-head pairs:	$(w_i, w_{\rho(i)}), (t_i, t_{\rho(i)})$
word-head-tag triples:	$(t_{\rho(i)}, w_i, t_i)$
	$(w_{\rho(i)}, w_i, t_i)$
	$(w_{\rho(i)}, t_{\rho(i)}, t_i)$
	$(w_{\rho(i)}, t_{\rho(i)}, w_i)$
tag-neighbourhood:	$(t_{\rho(i)}, t_{\rho(i)+1}, t_{i-1}, t_i)$
	$(t_{\rho(i)}, t_{\rho(i)+1}, t_{i+1}, t_i)$
	$(t_{\rho(i)}, t_{\rho(i)-1}, t_{i-1}, t_i)$
	$(t_{\rho(i)}, t_{\rho(i)-1}, t_{i+1}, t_i)$
between features:	$\forall_j\, i < j < \rho(i) \,\|\|\, \rho(i) < j < i$
	$(t_{\rho(i)}, t_j, t_i)$
arc-direction/length :	$(i - \rho(i) > 0, \|i - \rho(i)\|)$

3.2 Data and Tasks

In the next section, we present a set of scoring functions that can be used in the inline reranker loss framework, resulting in a new augmented-loss for each one. Augmented-loss learning is then applied to target a downstream task using the loss functions to measure gains. We show empirical results for two extrinsic loss-functions (optimizing for the downstream task): machine translation and domain adaptation; and for one intrinsic loss-function: an arc-length parsing score. For some experiments we also

measure the standard intrinsic parser metrics unlabeled attachment score (UAS) and labeled attachment score (LAS) (Buchholz and Marsi, 2006).

In terms of treebank data, the primary training corpus is the Penn Wall Street Journal Treebank (PTB) (Marcus et al., 1993). We also make use of the Brown corpus, and the Question Treebank (QTB) (Judge et al., 2006). For PTB and Brown we use standard training/development/testing splits of the data. For the QTB we split the data into three sections: 2000 training, 1000 development, and 1000 test. All treebanks are converted to dependency format using the Stanford converter v1.6 (de Marneffe et al., 2006).

4 Experiments

4.1 Machine Translation Reordering Score

As alluded to in Section 2.2, we use a reordering-based loss function to improve word order in a machine translation system. In particular, we use a system of source-side reordering rules which, given a parse of the source sentence, will reorder the sentence into a target-side order (Collins et al., 2005). In our experiments we work with a set of English-Japanese reordering rules[1] and gold reorderings based on human generated correct reordering of an aligned target sentences. We use a reordering score based on the reordering penalty from the METEOR scoring metric. Though we could have used a further downstream measure like BLEU, METEOR has also been shown to directly correlate with translation quality (Banerjee and Lavie, 2005) and is simpler to measure.

$$\text{reorder-score} = 1 - \frac{\#\text{ chunks} - 1}{\#\text{ unigrams\_matched} - 1}$$

$$\text{reorder-cost} = 1 - \text{reorder-score}$$

All reordering augmented-loss experiments are run with the same treebank data as the baseline (the training portions of PTB, Brown, and QTB). The extrinsic reordering training data consists of 10930 examples of English sentences and their correct Japanese word-order. We evaluate our results on an evaluation set of 6338 examples of similarly created reordering data. The reordering cost, evaluation criteria and data used in our experiments are based on the work of Talbot et al. (2011).

[1] Our rules are similar to those from Xu et al. (2009).

	Exact	Reorder
trans–PTB + Brown + QTB	35.29	76.49
trans–0.5×aug.-loss	38.71	78.19
trans–1.0×aug.-loss	39.02	78.39
trans–2.0×aug.-loss	39.58	78.67
graph–PTB + Brown + QTB	25.71	69.84
graph–0.5× aug.-loss	28.99	72.23
graph–1.0×aug.-loss	29.99	72.88
graph–2.0×aug.-loss	30.03	73.15

Table 1: Reordering scores for parser-based reordering (English-to-Japanese). Exact is the number of correctly reordered sentences. All models use the same treebank-data (PTB, QTB, and the Brown corpus). Results for three augmented-loss schedules are shown: 0.5 where for every two treebank updates we make one augmented-loss update, 1 is a 1-to-1 mix, and 2 is where we make twice as many augmented-loss updates as treebank updates.

Table 1 shows the results of using the reordering cost as an augmented-loss to the standard treebank objective function. Results are presented as measured by the reordering score as well as a coarse exact-match score (the number of sentences which would have correct word-order given the parse and the fixed reordering rules). We see continued improvements as we adjust the schedule to process the extrinsic loss more frequently, the best result being when we make two augmented-loss updates for every one treebank-based loss update.

4.2 Semi-supervised domain adaptation

Another application of the augmented-loss framework is to improve parser domain portability in the presence of partially labeled data. Consider, for example, the case of questions. Petrov et al. (2010) observed that dependency parsers tend to do quite poorly when parsing questions due to their limited exposure to them in the news corpora from the PennTreebank. Table 2 shows the accuracy of two parsers (LAS, UAS and the F1 of the root dependency attachment) on the QuestionBank test data. The first is a parser trained on the standard training sections of the PennTreebank (PTB) and the second is a parser trained on the training portion of the QuestionBank (QTB). Results for both

	LAS	UAS	Root-F1
trans–PTB	67.97	73.52	47.60
trans–QTB	84.59	89.59	91.06
trans–aug.-loss	76.27	86.42	83.41
graph–PTB	65.27	72.72	43.10
graph–QTB	82.73	87.44	91.58
graph–aug.-loss	72.82	80.68	86.26

Table 2: Domain adaptation results. Table shows (for both transition and graph-based parsers) the labeled accuracy score (LAS), unlabeled accuracy score (UAS) and Root-F1 for parsers trained on the PTB and QTB and tested on the QTB. The augmented-loss parsers are trained on the PTB but with a partial tree loss on QTB that considers only root dependencies.

transition-based parsers and graph-based parsers are given. Clearly there is significant drop in accuracy for a parser trained on the PTB. For example, the transition-based PTB parser achieves a LAS of 67.97% relative to 84.59% for the parser trained on the QTB.

We consider the situation where it is possible to ask annotators a single question about the target domain that is relatively easy to answer. The question should be posed so that the resulting answer produces a partially labeled dependency tree. Root-F1 scores from Table 2 suggest that one simple question is "what is the main verb of this sentence?" for sentences that are questions. In most cases this task is straight-forward and will result in a single dependency, that from the root to the main verb of the sentence. We feel this is a realistic partial labeled training setting where it would be possible to quickly collect a significant amount of data.

To test whether such weak information can significantly improve the parsing of questions, we trained an augmented-loss parser using the training set of the QTB stripped of all dependencies except the dependency from the root to the main verb of the sentence. In other words, for each sentence, the parser may only observe a single dependency at training from the QTB – the dependency to the main verb. Our augmented-loss function in this case is a simple binary function: 0 if a parse has the correct root dependency and 1 if it does not. Thus, the algorithm will select the first parse in the k-best list that has the correct root as the proxy to a gold standard parse.[2]

The last row in each section of Table 2 shows the results for this augmented-loss system when weighting both losses equally during training. By simply having the main verb annotated in each sentence – the sentences from the training portion of the QTB – the parser can eliminate half of the errors of the original parser. This is reflected by both the Root-F1 as well as LAS/UAS. It is important to point out that these improvements are not limited to simply better root predictions. Due to the fact that parsing algorithms make many parsing decisions jointly at test time, all such decisions influence each other and improvements are seen across the board. For example, the transition-based PTB parser has an F1 score of 41.22% for verb subjects (nsubj), whereas the augmented-loss parser has an F1 of 73.52%. Clearly improving just a single (and simple to annotate) dependency leads to general parser improvements.

4.3 Average Arc Length Score

The augmented-loss framework can be used to incorporate multiple treebank-based loss functions as well. Labeled attachment score is used as our base model loss function. In this set of experiments we consider adding an additional loss function which weights the lengths of correct and incorrect arcs, the average (labeled) arc-length score:

$$\text{ALS} = \frac{\sum_i \delta(\hat{\rho}_i, \rho_i)(i - \rho_i)}{\sum_i (i - \rho_i)}$$

For each word of the sentence we compute the distance between the word's position i and the position of the words head ρ_i. The arc-length score is the summed length of all those with correct head assignments ($\delta(\hat{\rho}_i, \rho_i)$ is 1 if the predicted head and the correct head match, 0 otherwise). The score is normalized by the summed arc lengths for the sentence. The labeled version of this score requires that the labels of the arc are also correct. Optimizing for dependency arc length is particularly important as parsers tend to do worse on longer dependencies (McDonald and Nivre, 2007) and these dependencies are typically the most meaningful for downstream tasks, e.g., main verb dependencies for tasks

[2] For the graph-based parser one can also find the higest scoring tree with correct root by setting the score of all competing arcs to $-\infty$.

	LAS	UAS	ALS
trans–PTB	88.64	91.64	82.96
trans–unlabeled aug.-loss	88.74	91.91	83.65
trans–labeled aug.-loss	88.84	91.91	83.46
graph–PTB	85.75	88.70	73.88
graph–unlabeled aug.-loss	85.80	88.81	74.26
graph–labeled aug.-loss	85.85	88.93	74.40

Table 3: Results for both parsers on the development set of the PTB. When training with ALS (labeled and unlabeled), we see an improvement in UAS, LAS, and ALS. Furthermore, if we use a labeled-ALS as the metric for augmented-loss training, we also see a considerable increase in LAS.

like information extraction (Yates and Etzioni, 2009) and textual entailment (Berant et al., 2010).

In Table 3 we show results for parsing with the ALS augmented-loss objective. For each parser, we consider two different ALS objective functions; one based on unlabeled-ALS and the other on labeled-ALS. The arc-length score penalizes incorrect long-distance dependencies more than local dependencies; long-distance dependencies are often more destructive in preserving sentence meaning and can be more difficult to predict correctly due to the larger context on which they depend. Combining this with the standard attachment scores biases training to focus on the difficult head dependencies.

For both experiments we see that by adding the ALS augmented-loss we achieve an improvement in LAS and UAS in addition to ALS. The augmented-loss not only helps us improve on the longer dependencies (as reflected in the increased ALS), but also in the main parser objective function of LAS and UAS. Using the labeled loss function provides better reinforcement as can be seen in the improvements over the unlabeled loss-function. As with all experiments in this paper, the graph-based parser baselines are much lower than the transition-based parser due to the use of arc-factored features. In these experiments we used an inline-ranker loss with 8 parses. We experimented with larger sizes (16 and 64) and found very similar improvements: for example, the transition parser's LAS for the labeled loss is 88.68 and 88.84, respectively).

We note that ALS can be decomposed locally and could be used as the primary objective function for parsing. A parse with perfect scores under ALS and LAS will match the gold-standard training tree. However, if we were to order incorrect parses of a sentence, ALS and LAS will suggest different orderings. Our results show that by optimizing for losses based on a combination of these metrics we train a more robust parsing model.

5 Related Work

A recent study by Katz-Brown et al. (2011) also investigates the task of training parsers to improve MT reordering. In that work, a parser is used to first parse a set of manually reordered sentences to produce k-best lists. The parse with the best reordering score is then fixed and added back to the training set and a new parser is trained on resulting data. The method is called *targeted self-training* as it is similar in vein to self-training (McClosky et al., 2006), with the exception that the new parse data is targeted to produce accurate word reorderings. Our method differs as it does not statically fix a new parse, but dynamically updates the parameters and parse selection by incorporating the additional loss in the inner loop of online learning. This allows us to give guarantees of convergence. Furthermore, we also evaluate the method on alternate extrinsic loss functions.

Liang et al. (2006) presented a perceptron-based algorithm for learning the phrase-translation parameters in a statistical machine translation system. Similar to the inline-ranker loss function presented here, they use a k-best lists of hypotheses in order to identify parameters which can improve a global objective function: BLEU score. In their work, they are interested in learning a parameterization over translation phrases (including the underlying word-alignment) which optimizes the BLEU score. Their goal is considerably different; they want to incorporate additional features into their model and define an objective function which allows them to do so; whereas, we are interested in allowing for multiple objective functions in order to adapt the parser model parameters to downstream tasks or alternative intrinsic (parsing) objectives.

The work that is most similar to ours is that of Chang et al. (2007), who introduced the Constraint Driven Learning algorithm (CODL). Their algorithm specifically optimizes a loss function with

the addition of constraints based on unlabeled data (what we call extrinsic datasets). For each unlabeled example, they use the current model along with their set of constraints to select a set of k automatically labeled examples which best meet the constraints. These induced examples are then added to their training set and, after processing each unlabeled dataset, they perform full model optimization with the concatenation of training data and newly generated training items. The augmented-loss algorithm can be viewed as an online version of this algorithm which performs model updates based on the augmented-loss functions directly (rather than adding a set of examples to the training set). Unlike the CODL approach, we do not perform complete optimization on each iteration over the unlabeled dataset; rather, we incorporate the updates in our online learning algorithm. As mentioned earlier, CODL is one example of learning algorithms that use weak supervision, others include Mann and McCallum (2010) and Ganchev et al. (2010). Again, these works are typically interested in using the extrinsic metric – or, in general, extrinsic information – to optimize the intrinsic metric in the absence of any labeled intrinsic data. Our goal is to optimize both simultaneously.

The idea of jointly training parsers to optimize multiple objectives is related to joint learning and inference for tasks like information extraction (Finkel and Manning, 2009) and machine translation (Burkett et al., 2010). In such works, a large search space that covers both the space of parse structures and the space of task-specific structures is defined and parameterized so that standard learning and inference algorithms can be applied. What sets our work apart is that there is still just a single parameter set that is being optimized – the parser parameters. Our method only uses feedback from task specific objectives in order to update the parser parameters, guiding it towards better downstream performance. This is advantageous for two reasons. First, it decouples the tasks, making inference and learning more efficient. Second, it does not force arbitrary paraemter factorizations in order to define a joint search space that can be searched efficiently.

Finally, augmented-loss training can be viewed as multi-task learning (Caruana, 1997) as the model optimizes multiple objectives over multiple data sets with a shared underlying parameter space.

6 Discussion

The empirical results show that incorporating an augmented-loss using the inline-ranker loss framework achieves better performance under metrics associated with the external loss function. For the intrinsic loss, we see that the augmented-loss framework can also result in an improvement in parsing performance; however, in the case of ALS, this is due to the fact that the loss function is very closely related to the standard evaluation metrics of UAS and LAS.

Although our analysis suggests that this algorithm is guaranteed to converge only for the separable case, it makes a further assumption that if there is a better parse under the augmented-loss, then there must be a lower cost parse in the k-best list. The empirical evaluation presented here is based on a very conservative approximation by choosing lists with at most 8 parses. However, in our experiments, we found that increasing the size of the lists did not significantly increase our accuracy under the external metrics. If we do have at least one improvement in our k-best lists, the analysis suggests that this is enough to move in the correct direction for updating the model. The assumption that there will always be an improvement in the k-best list if there is some better parse breaks down as training continues. We suspect that an increasing k, as suggested in Section 2.3, will allow for continued improvements.

Dependency parsing, as presented in this paper, is performed over (k-best) part-of-speech tags and is therefore dependent on the quality of the tagger. The experiments presented in this paper made use of a tagger trained on the source treebank data which severely limits the variation in parses. The augmented-loss perceptron algorithm presented here can be applied to any online learning problem, including part-of-speech tagger training. To build a dependency parser which is better adapted to a downstream task, one would want to perform augmented-loss training on the tagger as well.

7 Conclusion

We introduced the augmented-loss training algorithm and show that the algorithm can incorporate

additional loss functions to adapt the model towards extrinsic evaluation metrics. Analytical results are presented that show that the algorithm can optimize multiple objective functions simultaneously. We present an empirical analysis for training dependency parsers for multiple parsing algorithms and multiple loss functions.

The augmented-loss framework supports both intrinsic and extrinsic losses, allowing for both combinations of objectives as well as multiple sources of data for which the results of a parser can be evaluated. This flexibility makes it possible to tune a model for a downstream task. The only requirement is a metric which can be defined over parses of the downstream data. Our dependency parsing results show that we are not limited to increasing parser performance via more data or external domain adaptation techniques, but that we can incorporate the downstream task into parser training.

Acknowledgements: We would like to thank Kuzman Ganchev for feedback on an earlier draft of this paper as well as Slav Petrov for frequent discussions on this topic.

References

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

J. Berant, I. Dagan, and J. Goldberger. 2010. Global learning of focused entailment graphs. In *Proc. of ACL*.

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

D. Burkett, J. Blitzer, and D. Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Proc. of NAACL*.

R. Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

M.W. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of ACL*.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. 2010. Structured output learning with indirect supervision. In *Proc. of ICML*.

M. Collins, P. Koehn, and I. Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proc. of ACL*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of ACL*.

M.C. de Marneffe, B. MacCartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*, Genoa, Italy.

J.R. Finkel and C.D. Manning. 2009. Joint parsing and named entity recognition. In *Proc. of NAACL*.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

D. Gildea. 2001. Corpus variation and parser performance. In *Proc. of EMNLP*.

K. Hall. 2007. k-best spanning tree parsing. In *Proc. of ACL*, June.

J. Judge, A. Cahill, and J. Van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proc. of ACL*, pages 497–504.

J. Katz-Brown, S. Petrov, R. McDonald, D. Talbot, F. Och, H. Ichikawa, M. Seno, and H. Kazawa. 2011. Training a parser for machine translation reordering. In *Proc. of EMNLP*.

S. Kübler, R. McDonald, and J. Nivre. 2009. *Dependency parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

P. Liang, A. Bouchard-Ct, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of COLING/ACL*.

G.S. Mann and A. McCallum. 2010. Generalized Expectation Criteria for Semi-Supervised Learning with Weakly Labeled Data. *The Journal of Machine Learning Research*, 11:955–984.

M. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proc. of ACL*.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proc. of EMNLP-CoNLL*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

T. Nakagawa, K. Inui, and S. Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Proc. of NAACL*.

J. Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

S. Petrov, P.C. Chang, M. Ringgaard, and H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proc. of EMNLP*, pages 705–713.

D. Talbot, H. Kazawa, H. Ichikawa, J. Katz-Brown, M. Seno, and F. Och. 2011. A lightweight evaluation framework for machine translation reordering. In *Proc. of the Sixth Workshop on Statistical Machine Translation*.

M. Wang, N.A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.

P. Xu, J. Kang, M. Ringgaard, and F. Och. 2009. Using a dependency parser to improve SMT for Subject-Object-Verb languages. In *Proc. of NAACL*.

A. Yates and O. Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34(1):255–296.

Y. Zhang and S. Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In *Proc. of EMNLP*, pages 562–571.

Structured Sparsity in Structured Prediction

André F. T. Martins[*†] **Noah A. Smith**[*] **Pedro M. Q. Aguiar**[‡] **Mário A. T. Figueiredo**[†]
[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[‡]Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal
[†]Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal
{afm,nasmith}@cs.cmu.edu, aguiar@isr.ist.utl.pt, mtf@lx.it.pt

Abstract

Linear models have enjoyed great success in structured prediction in NLP. While a lot of progress has been made on efficient training with several loss functions, the problem of endowing learners with a mechanism for feature selection is still unsolved. Common approaches employ ad hoc filtering or L_1-regularization; both ignore the structure of the feature space, preventing practicioners from encoding structural prior knowledge. We fill this gap by adopting regularizers that promote *structured sparsity*, along with efficient algorithms to handle them. Experiments on three tasks (chunking, entity recognition, and dependency parsing) show gains in performance, compactness, and model interpretability.

1 Introduction

Models for structured outputs are in demand across natural language processing, with applications in information extraction, parsing, and machine translation. State-of-the-art models usually involve linear combinations of features and are trained discriminatively; examples are conditional random fields (Lafferty et al., 2001), structured support vector machines (Altun et al., 2003; Taskar et al., 2003; Tsochantaridis et al., 2004), and the structured perceptron (Collins, 2002a). In all these cases, the underlying optimization problems differ only in the choice of loss function; choosing among them has usually a small impact on predictive performance.

In this paper, we are concerned with *model selection*: which features should be used to define the prediction score? The fact that models with few features ("sparse" models) are desirable for several reasons (compactness, interpretability, good generalization) has stimulated much research work which has produced a wide variety of methods (Della Pietra et al., 1997; Guyon and Elisseeff, 2003; McCallum, 2003). Our focus is on methods which embed this selection into the learning problem via the regularization term. We depart from previous approaches in that we seek to make decisions jointly about all candidate features, and we want to promote sparsity patterns that go beyond the mere cardinality of the set of features. For example, we want to be able to select entire *feature templates* (rather than features individually), or to make the inclusion of some features depend on the inclusion of other features.

We achieve the goal stated above by employing regularizers which promote *structured sparsity*. Such regularizers are able to encode prior knowledge and guide the selection of features by modeling the structure of the feature space. Lately, this type of regularizers has received a lot of attention in computer vision, signal processing, and computational biology (Zhao et al., 2009; Kim and Xing, 2010; Jenatton et al., 2009; Obozinski et al., 2010; Jenatton et al., 2010; Bach et al., 2011). Eisenstein et al. (2011) employed structured sparsity in computational sociolinguistics. However, none of these works have addressed structured prediction. Here, we combine these two levels of structure: structure in the output space, and structure in the feature space. The result is a framework that allows building structured predictors with high predictive power, while reducing manual feature engineering. We obtain models that are interpretable, accurate, and often much more compact than L_2-regularized ones. Compared with L_1-regularized models, ours are often more accurate and yield faster runtime.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1500–1511,
Edinburgh, Scotland, UK, July 27–31, 2011.

2 Structured Prediction

We address structured prediction problems, which involve an input set $\mathcal{X}$ (*e.g.*, sentences) and an output set $\mathcal{Y}$, assumed large and structured (*e.g.*, tags or parse trees). We assume that each $x \in \mathcal{X}$ has a set of candidate outputs $\mathcal{Y}(x) \subseteq \mathcal{Y}$. We consider linear models, in which predictions are made according to

$$\widehat{y} = \arg\max_{y \in \mathcal{Y}(x)} \boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, y), \quad (1)$$

where $\boldsymbol{\phi}(x, y) \in \mathbb{R}^D$ is a vector of features, and $\boldsymbol{\theta} \in \mathbb{R}^D$ is the vector of corresponding weights. Let $\mathcal{D} = \{\langle x_i, y_i \rangle\}_{i=1}^N$ be a training sample. We assume a cost function is defined such that $c(\widehat{y}, y)$ is the cost of predicting $\widehat{y}$ when the true output is y; our goal is to learn $\boldsymbol{\theta}$ with small expected cost on unseen data. To achieve this goal, linear models are usually trained by solving a problem of the form

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta}) + \tfrac{1}{N} \textstyle\sum_{i=1}^N L(\boldsymbol{\theta}, x_i, y_i), \quad (2)$$

where Ω is a *regularizer* and L is a *loss* function. Examples of losses are: the negative conditional log-likelihood used in CRFs (Lafferty et al., 2001),

$$L_{\text{CRF}}(\boldsymbol{\theta}, x, y) = -\log P_{\boldsymbol{\theta}}(y|x), \quad (3)$$

where $P_{\boldsymbol{\theta}}(y|x) \propto \exp(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, y))$ is a log-linear model; the margin rescaled loss of structured SVMs (Taskar et al., 2003; Tsochantaridis et al., 2004),

$$L_{\text{SVM}}(\boldsymbol{\theta}, x, y) = \max_{y' \in \mathcal{Y}(x)} \boldsymbol{\theta} \cdot \delta\boldsymbol{\phi}(y') + c(y', y), \quad (4)$$

where $\delta\boldsymbol{\phi}(y') = \boldsymbol{\phi}(x, y') - \boldsymbol{\phi}(x, y)$; and the loss underlying the structured perceptron (Collins, 2002a),

$$L_{\text{SP}}(\boldsymbol{\theta}, x, y) = \max_{y' \in \mathcal{Y}(x)} \boldsymbol{\theta} \cdot \delta\boldsymbol{\phi}(y'). \quad (5)$$

Empirical comparison among these loss functions can be found in the literature (see, *e.g.*, Martins et al., 2010, who also consider interpolations of the losses above). In practice, it has been observed that the choice of loss has far less impact than the model design and choice of features. Hence, in this paper, we focus our attention on the regularization term in Eq. 2. We specifically address ways in which this term can be used to help design the model by promoting *structured sparsity*. While this has been a topic of intense research in signal processing and computational biology (Jenatton et al., 2009; Liu and Ye, 2010; Bach et al., 2011), it has not yet received much attention in the NLP community, where the choice of regularization for supervised learning has essentially been limited to the following:

- L_2-regularization (Chen and Rosenfeld, 2000):

$$\Omega_\lambda^{L_2}(\boldsymbol{\theta}) \triangleq \tfrac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 = \tfrac{\lambda}{2}\textstyle\sum_{d=1}^D \theta_d^2; \quad (6)$$

- L_1-regularization (Kazama and Tsujii, 2003; Goodman, 2004):

$$\Omega_\tau^{L_1}(\boldsymbol{\theta}) \triangleq \tau\|\boldsymbol{\theta}\|_1 = \tau\textstyle\sum_{d=1}^D |\theta_d|. \quad (7)$$

The latter is known as "Lasso," as popularized by Tibshirani (1996) in the context of sparse regression. In the two cases above, λ and τ are nonnegative coefficients controlling the intensity of the regularization. $\Omega_\lambda^{L_2}$ usually leads to easier optimization and robust performance; $\Omega_\tau^{L_1}$ encourages sparser models, where only a few features receive nonzero weights; see Gao et al. (2007) for an empirical comparison. More recently, Petrov and Klein (2008b) applied L_1 regularization for structure learning in phrase-based parsing; a comparison with L_2 appears in Petrov and Klein (2008a). Elastic nets interpolate between L_1 and L_2, having been proposed by Zou and Hastie (2005) and used by Lavergne et al. (2010) to regularize CRFs.

Neither of the regularizers just described "looks" at the *structure* of the feature space, since they all treat each dimension independently—we call them *unstructured* regularizers, as opposed to the structured ones that we next describe.

3 Structured Sparsity

We are interested in regularizers that share with $\Omega_\tau^{L_1}$ the ability to promote sparsity, so that they can be used for selecting features. In addition, we want to endow the feature space $\mathbb{R}^D$ with additional structure, so that features are not penalized individually (as in the L_1-case) but collectively, encouraging entire *groups* of features to be discarded. The choice of groups will allow encoding prior knowledge regarding the kind of sparsity patterns that are intended in the model. This can be achieved with *group-Lasso regularization*, which we next describe.

3.1 The Group Lasso

To capture the structure of the feature space, we group our D features into M groups $G_1, \ldots, G_M$, where each $G_m \subseteq \{1, \ldots, D\}$. Ahead, we discuss meaningful ways of choosing group decompositions; for now, let us assume a sensible choice is obvious to the model designer. Denote by $\boldsymbol{\theta}_m = \langle \theta_d \rangle_{d \in G_m}$ the subvector of those weights that correspond to the features in the m-th group, and let $d_1, \ldots, d_M$ be nonnegative scalars (one per group). We consider the following *group-Lasso* regularizers:

$$\Omega^{\mathrm{GL}}_{\boldsymbol{d}} = \sum_{m=1}^{M} d_m \|\boldsymbol{\theta}_m\|_2. \quad (8)$$

These regularizers were first proposed by Bakin (1999) and Yuan and Lin (2006) in the context of regression. If $d_1 = \ldots = d_M$, $\Omega^{\mathrm{GL}}_{\boldsymbol{d}}$ becomes the "L_1 norm of the L_2 norms." Interestingly, this is also a norm, called the mixed $L_{2,1}$-norm.[1] These regularizers subsume the L_1 and L_2 cases, which correspond to trivial choices of groups:

- If each group is a singleton, *i.e.*, $M = D$ and $G_d = \{\theta_d\}$, and $d_1 = \ldots = d_M = \tau$, we recover L_1-regularization (cf. Eqs. 7–8).
- If there is a single group spanning all the features, *i.e.*, $M = 1$ and $G_1 = \{1, \ldots, D\}$, then the right hand side of Eq. 8 becomes $d_1 \|\boldsymbol{\theta}\|_2$. This is equivalent to L_2 regularization.[2]

We next present some non-trivial examples concerning different topologies of $\mathcal{G} = \{G_1, \ldots, G_M\}$.

Non-overlapping groups. Let us first consider the case where $\mathcal{G}$ is a *partition* of the feature space: the groups cover all the features ($\bigcup_m G_m = \{1, \ldots, D\}$), and they do not overlap ($G_a \cap G_b = \varnothing$, $\forall a \neq b$). Then, $\Omega^{\mathrm{GL}}_{\boldsymbol{d}}$ is termed a *non-overlapping group-Lasso* regularizer. It encourages sparsity patterns in which entire groups are discarded. A judicious choice of groups can lead to very compact models and pinpoint relevant groups of features. The following examples lie in this category:

- The two cases above (L_1 and L_2 regularization).
- *Label-based groups*. In multi-label classification, where $\mathcal{Y} = \{1, \ldots, L\}$, features are typically designed as conjunctions of input features with label indicators, i.e., they take the form $\boldsymbol{\phi}(x, y) = \boldsymbol{\psi}(x) \otimes \boldsymbol{e}_y$, where $\boldsymbol{\psi}(x) \in \mathbb{R}^{D_X}$, $\boldsymbol{e}_y \in \mathbb{R}^L$ has all entries zero except the y-th entry, which is 1, and $\otimes$ denotes the Kronecker product. Hence $\boldsymbol{\phi}(x, y)$ can be reshaped as a D_X-by-L matrix, and we can let each group correspond to a row. In this case, all groups have the same size and we typically set $d_1 = \ldots = d_M$. A similar design can be made for sequence labeling problems, by considering a similar grouping for the unigram features.[3]
- *Template-based groups*. In NLP, features are commonly designed via *templates*. For example, a template such as $w_0 \wedge p_0 \wedge p_{-1}$ denotes the word in the current position (w_0) conjoined with its part-of-speech (p_0) and that of the previous word (p_{-1}). This template encloses many features corresponding to different instantiantions of w_0, p_0, and p_{-1}. In §5, we learn *feature templates* from the data, by associating each group to a feature template, and letting that group contain all features that are instantiations of this template. Since groups have different sizes, it is a good idea to let d_m increase with the group size, so that larger groups pay a larger penalty for being included.

Tree-structured groups. More generally, we may let the groups in $\mathcal{G}$ overlap but be nested, *i.e.*, we may want them to form a *hierarchy* (two distinct groups either have empty intersection or one is contained in the other). This induces a partial order on $\mathcal{G}$ (the set inclusion relation $\supseteq$), endowing it with the structure of a partially ordered set (*poset*).

A convenient graphical representation of the poset $\langle \mathcal{G}, \supseteq \rangle$ is its *Hasse diagram*. Each group is a node in the diagram, and an arc is drawn from group G_a to group G_b if $G_b \subset G_a$ and there is no b' s.t. $G_b \subset G_{b'} \subset G_a$. When the groups are nested, this diagram is a *forest* (a union of directed trees). The corresponding regularizer enforces sparsity patterns

[1] In the statistics literature, such mixed-norm regularizers, which group features and then apply a separate norm for each group, are called *composite absolute penalties* (Zhao et al., 2009); other norms besides $L_{2,1}$ can be used, such as $L_{\infty,1}$ (Quattoni et al., 2009; Wright et al., 2009; Eisenstein et al., 2011).

[2] Note that Eqs. 8 and 6 do not become *exactly* the same: in Eq. 6, the L_2 norm is squared. However it can be shown that both regularizers lead to identical learning problems (Eq. 2) up to a transformation of the regularization constant.

[3] The same idea is also used in multitask learning, where labels correspond to tasks (Caruana, 1997).

where a group of features is only selected if *all its ancestors are also selected*.[4] Hence, entire subtrees in the diagram can be pruned away. Examples are:

- The *elastic net*. The diagram of $\mathcal{G}$ has a root node for $G_1 = \{1, \ldots, D\}$ and D leaf nodes, one per each singleton group (see Fig. 1).
- The *sparse group-Lasso*. This regularizer was proposed by Friedman et al. (2010):

$$\Omega^{\mathrm{SGL}}_{\boldsymbol{d},\boldsymbol{\tau}}(\boldsymbol{\theta}) = \sum_{m=1}^{M'} \left(d_m \|\boldsymbol{\theta}_m\|_2 + \tau_m \|\boldsymbol{\theta}_m\|_1\right), \tag{9}$$

  where the total number of groups is $M = M' + D$, and the components $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{M'}$ are non-overlapping. This regularizer promotes sparsity at both group and feature levels (*i.e.*, it eliminates entire groups and sparsifies within each group).

Graph-structured groups. In general, the groups in $\mathcal{G}$ may overlap without being nested. In this case, the Hasse diagram of $\mathcal{G}$ is a directed acyclic graph (DAG). As in the tree-structured case, a group of features is only selected if all its ancestors are also selected. Based on this property, Jenatton et al. (2009) suggested a way of reverse engineering the groups from the desired sparsity pattern. We next describe a strategy for *coarse-to-fine feature template selection* that directly builds on that idea.

Suppose that we are given M feature templates $\mathcal{T} = \{T_1, \ldots, T_M\}$ which are partially ordered according to some criterion, such that if $T_a \preceq T_b$ we would like to include T_b in our model only if T_a is also included. This criterion could be a measure of coarseness: we may want to let coarser part-of-speech features precede finer lexical features, *e.g.*, $p_0 \wedge p_1 \preceq w_0 \wedge w_1$, or conjoined features come after their elementary parts, *e.g.*, $p_0 \preceq p_0 \wedge p_1$. The order does not need to be total, so some templates may not be comparable (*e.g.*, we may want $p_0 \wedge p_{-1}$ and $p_0 \wedge p_1$ not to be comparable). To achieve the sparsity pattern encoded in $\langle \mathcal{T}, \preceq \rangle$, we choose $\mathcal{G} = \langle G_1, \ldots, G_M \rangle$ as follows: let $I(T_a)$ be the set of features that are instantiations of template T_a; then define $G_a = \bigcup_{b:a \preceq b} I(T_b)$, for $a = 1, \ldots, M$. It is easy to see that $\langle \mathcal{G}, \supseteq \rangle$ and $\langle \mathcal{T}, \preceq \rangle$ are isomorph posets (their Hasse diagrams have the same shape; see Fig. 1). The result is a "coarse-to-fine" regularizer, which prefers to select feature templates that are coarser before zooming into finer features.

[4] We say that a group of features G_m is selected if *some* feature in G_m (but not necessarily all) has a nonzero weight.

3.2 Bayesian Interpretation

The prior knowledge encoded in the group-Lasso regularizer (Eq. 8) comes with a Bayesian interpretation, as we next describe. In a probabilistic model (e.g. in the CRF case, where $L = L_{\mathrm{CRF}}$), the optimization problem in Eq. 2 can be seen as maximum *a posteriori* estimation of $\boldsymbol{\theta}$, where the regularization term $\Omega(\boldsymbol{\theta})$ corresponds to the negative log of a prior distribution (call it $p(\boldsymbol{\theta})$). It is well-known that L_2-regularization corresponds to choosing independent zero-mean Gaussian priors, $\theta_d \sim \mathcal{N}(0, \lambda^{-1})$, and that L_1-regularization results from adopting zero-mean Laplacian priors, $p(\theta_d) \propto \exp(\tau|\theta_d|)$.

Figueiredo (2002) provided an alternative interpretation of L_1-regularization in terms of a two-level hierarchical Bayes model, which happens to generalize to the non-overlapping group-Lasso case, where $\Omega = \Omega^{\mathrm{GL}}_{\boldsymbol{d}}$. As in the L_2-case, we also assume that each parameter receives a zero-mean Gaussian prior, but now with a *group-specific variance* τ_m, *i.e.*, $\boldsymbol{\theta}_m \sim \mathcal{N}(\mathbf{0}, \tau_m \boldsymbol{I})$ for $m = 1, \ldots, M$. This reflects the fact that some groups should have their feature weights shrunk more towards zero than others. The variances $\tau_m \geq 0$ are not pre-specified but rather generated by a one-sided exponential hyperprior $p(\tau_m|d_m) \propto \exp(-d_m^2 \tau_m/2)$. It can be shown that after marginalizing out τ_m, we obtain

$$\begin{aligned} p(\boldsymbol{\theta}_m|d_m) &= \int_0^\infty p(\boldsymbol{\theta}_m|\tau_m) p(\tau_m|d_m) d\tau_m \\ &\propto \exp\left(-d_m \|\boldsymbol{\theta}_m\|\right). \end{aligned} \tag{10}$$

Hence, the non-overlapping group-Lasso corresponds to the following two-level hierachical Bayes model: independently for each $m = 1, \ldots, M$,

$$\tau_m \sim \mathrm{Exp}(d_m^2/2), \qquad \boldsymbol{\theta}_m \sim \mathcal{N}(0, \tau_m \boldsymbol{I}). \tag{11}$$

3.3 Prox-operators

Before introducing our learning algorithm for handling group-Lasso regularization, we need to define the concept of a Ω-*proximity operator*. This is the function $\mathrm{prox}_\Omega : \mathbb{R}^D \to \mathbb{R}^D$ defined as follows:

$$\mathrm{prox}_\Omega(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}'} \tfrac{1}{2}\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|^2 + \Omega(\boldsymbol{\theta}'). \tag{12}$$

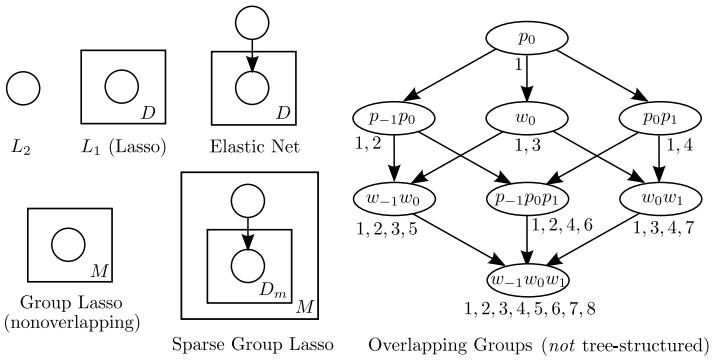


Figure 1: Hasse diagrams of several group-based regularizers. For all tree-structured cases, we use the same plate notation that is traditionally used in probabilistic graphical models. The rightmost diagram represents a coarse-to-fine regularizer: each node is a template involving contiguous sequences of words (w) and POS tags (p); the symbol order $\varnothing \preceq p \preceq w$ induces a template order ($T_a \preceq T_b$ iff at each position i $[T_a]_i \preceq [T_b]_i$). Digits below each node are the group indices where each template belongs.

Proximity operators generalize Euclidean projections and have many interesting properties; see Bach et al. (2011) for an overview. By requiring zero to be a subgradient of the objective function in Eq. 12, we obtain the following closed expression (called *soft-thresholding*) for the $\Omega_\tau^{L_1}$-proximity operator:

$$[\mathrm{prox}_{\Omega_\tau^{L_1}}(\boldsymbol{\theta})]_d = \begin{cases} \theta_d - \tau & \text{if } \theta_d > \tau \\ 0 & \text{if } |\theta_d| \leq \tau \\ \theta_d + \tau & \text{if } \theta_d < -\tau. \end{cases} \quad (13)$$

For the *non-overlapping* group Lasso case, the proximity operator is given by

$$[\mathrm{prox}_{\Omega_{\boldsymbol{d}}^{\mathrm{GL}}}(\boldsymbol{\theta})]_m = \begin{cases} \mathbf{0} & \text{if } \|\boldsymbol{\theta}_m\|_2 \leq d_m \\ \frac{\|\boldsymbol{\theta}_m\|_2 - d_m}{\|\boldsymbol{\theta}_m\|_2}\boldsymbol{\theta}_m & \text{otherwise.} \end{cases} \quad (14)$$

which can be seen as a generalization of Eq. 13: if the L_2-norm of the m-th group is less than d_m, the entire group is discarded; otherwise it is scaled so that its L_2-norm decreases by an amount of d_m.

When groups overlap, the proximity operator lacks a closed form. When $\mathcal{G}$ is tree-structured, it can still be efficiently computed by a recursive procedure (Jenatton et al., 2010). When $\mathcal{G}$ is *not* tree-structured, no specialized procedure is known, and a convex optimizer is necessary to solve Eq. 12.

4 Online Prox-Grad Algorithm

We now turn our attention to efficient ways of handling group-Lasso regularizers. Several fast and scalable algorithms having been proposed for training L_1-regularized CRFs, based on quasi-Newton optimization (Andrew and Gao, 2007), coordinate descent (Sokolovska et al., 2010; Lavergne et al., 2010), and stochastic gradients (Carpenter, 2008; Langford et al., 2009; Tsuruoka et al., 2009). The algorithm that we use in this paper (Alg. 1) extends the stochastic gradient methods for group-Lasso regularization; a similar algorithm was used by Martins et al. (2011) for multiple kernel learning.

Alg. 1 addresses the learning problem in Eq. 2 by alternating between online (sub-)gradient steps with respect to the loss term, and proximal steps with respect to the regularizer. Proximal-gradient methods are very popular in sparse modeling, both in batch (Liu and Ye, 2010; Bach et al., 2011) and online (Duchi and Singer, 2009; Xiao, 2009) settings. The reason we have chosen the algorithm of Martins et al. (2011) is that it effectively handles overlapping groups, without the need of evaluating prox_Ω (which, as seen in §3.3, can be costly if $\mathcal{G}$ is not tree-structured). To do so, it decomposes Ω as

$$\Omega(\boldsymbol{\theta}) = \textstyle\sum_{j=1}^{J} \sigma_j \Omega_j(\boldsymbol{\theta}) \quad (15)$$

for some $J \geq 1$, and nonnegative $\sigma_1, \ldots, \sigma_J$; each Ω_j-proximal operator is assumed easy to compute. Such a decomposition always exists: if $\mathcal{G}$ does not have overlapping groups, take $J = 1$. Otherwise, find $J \leq M$ disjoint sets $\mathcal{G}_1, \ldots, \mathcal{G}_J$ such that $\bigcup_{j=1}^{J} \mathcal{G}_j = \mathcal{G}$ and the groups on each $\mathcal{G}_j$ are non-overlapping. The proximal steps are then applied sequentially, one per each Ω_j. Overall, Alg. 1 satisfies the following important requirements:

- *Computational efficiency.* Each gradient step at round t is *linear* in the number of features that fire for that instance and *independent* of the total number of features D. Each proximal step is *linear* in the number of groups M, and does not need be to performed every round (as we will see later).

Algorithm 1 Online Sparse Prox-Grad Algorithm

input: $\mathcal{D}$, $\langle \Omega_j \rangle_{j=1}^J$, T, gravity sequence $\langle \langle \sigma_{jt} \rangle_{j=1}^J \rangle_{t=1}^T$, stepsize sequence $\langle \eta_t \rangle_{t=1}^T$
initialize $\boldsymbol{\theta} = \mathbf{0}$
for $t = 1$ **to** T **do**
  take training pair $\langle x_t, y_t \rangle \in \mathcal{D}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_t \nabla L(\boldsymbol{\theta}; x_t, y_t)$ (gradient step)
  for $j = 1$ **to** J **do**
    $\boldsymbol{\theta} = \mathrm{prox}_{\eta_t \sigma_{jt} \Omega_j}(\boldsymbol{\theta})$ (proximal step)
  end for
end for
output: $\boldsymbol{\theta}$

- *Memory efficiency.* Only a small active set of features (those that have nonzero weights) need to be maintained. Entire groups of features can be deleted after each proximal step. Furthermore, only the features which correspond to nonzero entries in the gradient vector need to be inserted in the active set; for some losses (L_{SVM} and L_{SP}) many irrelevant features are never instantianted.
- *Convergence.* With high probability, Alg. 1 produces an ϵ-accurate solution after $T \leq O(1/\epsilon^2)$ rounds, for a suitable choice of stepsizes and holding σ_{jt} constant, $\sigma_{jt} = \sigma_j$ (Martins et al., 2011). This result can be generalized to any sequence $\langle \sigma_{jt} \rangle_{t=1}^T$ such that $\sigma_j = \frac{1}{T} \sum_{t=1}^T \sigma_{jt}$.

We next describe several algorithmic ingredients that make Alg. 1 effective in sparse modeling.

Budget-Driven Shrinkage. Alg. 1 requires the choice of a "gravity sequence." We follow Langford et al. (2009) and set $\langle \sigma_{jt} \rangle_{j=1}^J$ to zero for all t which is not a multiple of some prespecified integer K; this way, proximal steps need only be performed each K rounds, yielding a significant speed-up when the number of groups M is large. A direct adoption of the method of Langford et al. (2009) would set $\sigma_{jt} = K\sigma_j$ for those rounds; however, we have observed that such a strategy makes the number of groups vary substantially in early epochs. We use a different strategy: for each $\mathcal{G}_j$, we specify a *budget* of $B_j \geq 0$ groups (this may take into consideration practical limitations, such as the available memory). If t is a multiple of K, we set σ_{jt} as follows:

1. If $\mathcal{G}_j$ does not have more than B_j nonzero groups, set $\sigma_{jt} = 0$ and do nothing.
2. Otherwise, sort the groups in $\mathcal{G}_j$ by decreasing order of their L_2-norms. Check the L_2-norms of the B_j-th and B_{j+1}-th entries in the list and set σ_{jt} as the mean of these two divided by η_t.
3. Apply a $\eta_t \sigma_{jt} \Omega_j$-proximal step using Eq. 14. At the end of this step, no more than B_j groups will remain nonzero.[5]

If the average of the gravity steps converge, $\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^T \sigma_{jt} \to \sigma_j$, then the limit points σ_j implicitly define the regularizer, via $\Omega = \sum_{j=1}^J \sigma_j \Omega_j$.[6] Hence, we have shifted the control of the amount of regularization to the budget constants B_j, which unlike the σ_j have a clear meaning and can be chosen under practical considerations.

Space and Time Efficiency. The proximal steps in Alg. 1 have a *scaling* effect on each group, which affects all features belonging to that group (see Eq. 14). We want to avoid explicitly updating each feature in the active set, which could be time consuming. We mention two strategies that can be used for the *non-overlapping* group Lasso case.

- The first strategy is suitable when M is large and only a few groups ($\ll M$) have features that fire in each round; this is the case, *e.g.*, of *label-based groups* (see §3.1). It consists of making *lazy updates* (Carpenter, 2008), *i.e.*, to delay the update of all features in a group until at least one of them fires; then apply a cumulative penalty. The amount of the penalty can be computed if one assigns a timestamp to each group.
- The second strategy is suitable when M is small and some groups are very populated; this is the typical case of *template-based groups* (§3.1). Two operations need to be performed: updating each feature weight (in the gradient steps), and scaling entire groups (in the proximal steps). We adapt a trick due to Shalev-Shwartz et al. (2007): represent the weight vector of the m-th group, $\boldsymbol{\theta}_m$, by a

[5] When overlaps exist (e.g. the coarse-to-fine case), we specify a total pseudo-budget B ignoring the overlaps, which induces budgets $B_1, \ldots, B_J$ which sum to B. The number of actually selected groups may be less than B, however, since in this case some groups can be shrunk more than once. Other heuristics are possible.

[6] The convergence assumption can be sidestepped by freezing the σ_j after a fixed number of iterations.

triple $\langle \boldsymbol{\xi}_m, c_m, \rho_m \rangle \in \mathbb{R}^{|G_m|} \times \mathbb{R}_+ \times \mathbb{R}_+$, such that $\boldsymbol{\theta}_m = c_m \boldsymbol{\xi}_m$ and $\|\boldsymbol{\theta}_m\|^2 = \rho_m$. This representation allows performing the two operations above in constant time, and it keeps track of the group L_2-norms, necessary in the proximal updates.

For sufficient amounts of regularization, our algorithm has a low memory footprint. Only features that, at some point, intervene in the gradient computed in line 5 need to be instantiated; and all features that receive zero weights after some proximal step can be deleted from the model (cf. Fig. 2).

Sparseptron and Debiasing. Although Alg. 1 allows to simultaneously select features and learn the model parameters, it has been observed in the sparse modeling literature that Lasso-like regularizers usually have a strong bias which may harm predictive performance. A post-processing stage is usually taken (called *debiasing*), in which the model is refitted without any regularization and using only the selected features (Wright et al., 2009). If a final debiasing stage is to be performed, Alg. 1 only needs to worry about feature selection, hence it is appealing to choose a loss function that makes this procedure as simple as possible. Examining the input of Alg. 1, we see that both a gravity and a stepsize sequence need to be specified. The former can be taken care of by using budget-driven shrinkage, as described above. The stepsize sequence can be set as $\eta_t = \eta_0 / \sqrt{\lceil t/N \rceil}$, which ensures convergence, however η_0 requires tuning. Fortunately, for the structured perceptron loss L_{SP} (Eq. 5), Alg. 1 is independent of η_0, up to a scaling of $\boldsymbol{\theta}$, which does not affect predictions (see Eq. 1).[7] We call the instantiation of Alg. 1 with a group-Lasso regularizer and the loss L_{SP} the *sparseptron*. Overall, we propose the following two-stage approach:

1. Run the sparsepton for a few epochs and discard the features with zero weights.
2. Refit the model without any regularization and using the loss L which one wants to optimize.

[7] To see why this is the case, note that both gradient and proximal updates come scaled by η_0; and that the gradient of the loss is $\nabla L_{\mathrm{SP}}(\boldsymbol{\theta}, x_t, y_t) = \boldsymbol{\phi}(x_t, \widehat{y}_t) - \boldsymbol{\phi}(x_t, y_t)$, where $\widehat{y}_t$ is the prediction under the current model, which is insensitive to the scaling of $\boldsymbol{\theta}$. This independence on η_0 does not hold when the loss is L_{SVM} or L_{CRF}.

5 Experiments

We present experiments in three structured prediction tasks for several group choices.

Text Chunking. We use the English dataset provided in the CoNLL 2000 shared task (Sang and Buchholz, 2000), which consists of 8,936 training and 2,012 testing sentences (sections 15–18 and 20 of the WSJ.) The input observations are the token words and their POS tags; we want to predict the sequences of `IOB` tags representing phrase chunks. We built 96 contextual feature templates as follows:

- Up to 5-grams of POS tags, in windows of 5 tokens on the left and 5 tokens on the right;
- Up to 3-grams of words, in windows of 3 tokens on the left and 3 tokens on the right;
- Up to 2-grams of word shapes, in windows of 2 tokens on the left and 2 tokens on the right. Each shape replaces characters by their types (case sensitive letters, digits, and punctuation), and deletes repeated types—*e.g.*, `Confidence` and `2,664,098` are respectively mapped to `Aa` and `0,0+,0+` (Collins, 2002b).

We defined unigram features by conjoining these templates with each of the 22 output labels. An additional template was defined to account for label bigrams—features in this template do not look at the input string, but only at consecutive pairs of labels.[8]

We evaluate the ability of group-Lasso regularization to perform *feature template selection*. To do that, we ran 5 epochs of the sparseptron algorithm with template-based groups and budget-driven shrinkage (budgets of 10, 20, 30, 40, and 50 templates were tried). For each group $\mathcal{G}_m$, we set $d_m = \log_2 |G_m|$, which is the average number of bits necessary to encode a feature in that group, if all features were equiprobable. We set $K = 1000$ (the number of instances between consecutive proximal steps). Then, we refit the model with 10 iterations of the max-loss 1-best MIRA algorithm (Crammer et al., 2006).[9] Table 1 compares the F_1 scores and

[8] State-of-the-art models use larger output contexts, such as label trigrams and 4-grams. We resort to bigram labels as we are mostly interested in identifying relevant unigram templates.

[9] This variant optimizes the L_{SVM} loss (Martins et al., 2010). For the refitting, we used unregularized MIRA. For the baseline

Table 1: Results for text chunking.

	MIRA	Group Lasso $B = 10$	$B = 20$	$B = 30$	$B = 40$	$B = 50$
F_1 (%)	93.10	92.99	93.28	93.59	93.42	93.40
model size (# features)	5,300,396	71,075	158,844	389,065	662,018	891,378

	MIRA	Lasso $C = 0.1$	$C = 0.5$	$C = 1$	Group-Lasso $B = 100$	$B = 200$	$B = 300$
Spa. dev/test	70.38/74.09	69.19/71.9	70.75/72.38	71.7/74.03	71.79/73.62	72.08/75.05	71.48/73.3
	8,598,246	68,565	1,017,769	1,555,683	83,036	354,872	600,646
Dut. dev/test	69.15/71.54	64.07/66.35	66.82/69.42	70.43/71.89	69.48/72.83	71.03/73.33	71.2/72.59
	5,727,004	164,960	565,704	953,668	128,320	447,193	889,660
Eng. dev/test	83.95/79.81	80.92/76.95	82.58/78.84	83.38/79.35	85.62/80.26	85.86/81.47	85.03/80.91
	8,376,901	232,865	870,587	1,114,016	255,165	953,178	1,719,229

Table 2: Results for named entity recognition. Each cell shows F_1 (%) and the number of features.

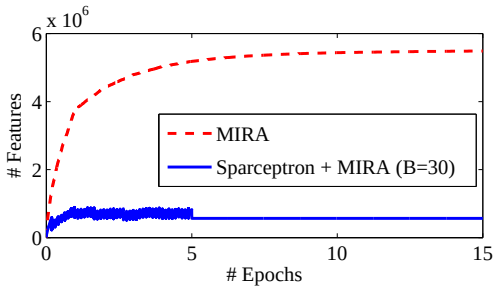


Figure 2: Memory footprints of the MIRA and sparseptron algorithms in text chunking. The oscillation in the first 5 epochs (bottom line) comes from the proximal steps each $K = 1000$ rounds. The features are then frozen and 10 epochs of unregularized MIRA follow. Overall, the sparseptron requires $< 7.5\%$ of the memory as the MIRA baseline.

the model sizes obtained with the several budgets against those obtained by running 15 iterations of MIRA with the original set of features. Note that the total number of iterations is the same; yet, the group-Lasso approach has a much smaller memory footprint (see Fig. 2) and yields much more compact models. The small memory footprint comes from the fact that Alg. 1 may entertain a large number of features without ever instantiating all of them. The predictive power is comparable (although some choices of budget yield slightly better scores for the group-Lasso approach).[10]

Named Entity Recognition. We experiment with the Spanish, Dutch, and English datasets provided in the CoNLL 2002/2003 shared tasks (Sang, 2002; Sang and De Meulder, 2003). For Spanish, we use the POS tags provided by Carreras (`http://www.lsi.upc.es/~nlp/tools/nerc/nerc.html`); for English, we ignore the syntactic chunk tags provided with the dataset. Hence, all datasets have the same sort of input observations (words and POS) and all have 9 output labels. We use the feature templates described above plus some additional ones (yielding a total of 452 templates):

- Up to 3-grams of shapes, in windows of size 3;
- For prefix/suffix sizes of $1, 2, 3$, up to 3-grams of word prefixes/suffixes, in windows of size 3;
- Up to 5-grams of case, punctuation, and digit indicators, in windows of size 5.

As before, an additional feature template was defined to account for label bigrams. We do feature template selection (same setting as before) for budget sizes of 100, 200, and 300. We compare with both MIRA (using all the features) and the sparseptron with a standard Lasso regularizer $\Omega_\tau^{L_1}$, for several values of $C = 1/(\tau N)$. Table 2 shows the results. We observe that template-based group-Lasso wins both in terms of accuracy and compactness. Note also that the ability to discard feature *templates* (rather than individual features) yields faster test runtime than models regularized with the standard Lasso: fewer templates will need to be instantiated, with a speed-up in score computation.

Multilingual Dependency Parsing. We trained non-projective dependency parsers for 6 languages using the CoNLL-X shared task datasets (Buchholz and Marsi, 2006): Arabic, Danish, Dutch, Japanese, Slovene, and Spanish. We chose the languages with the smallest datasets, because regularization is more important when data is scarce. The output to be predicted from each input sentence is the set of dependency links, which jointly define a spanning tree.

(described next), we used L_2-regularized MIRA and tuned the regularization constant with cross-validation.

[10] We also tried label-based group-Lasso and sparse group-Lasso (§3.1), with less impressive results (omitted for space).

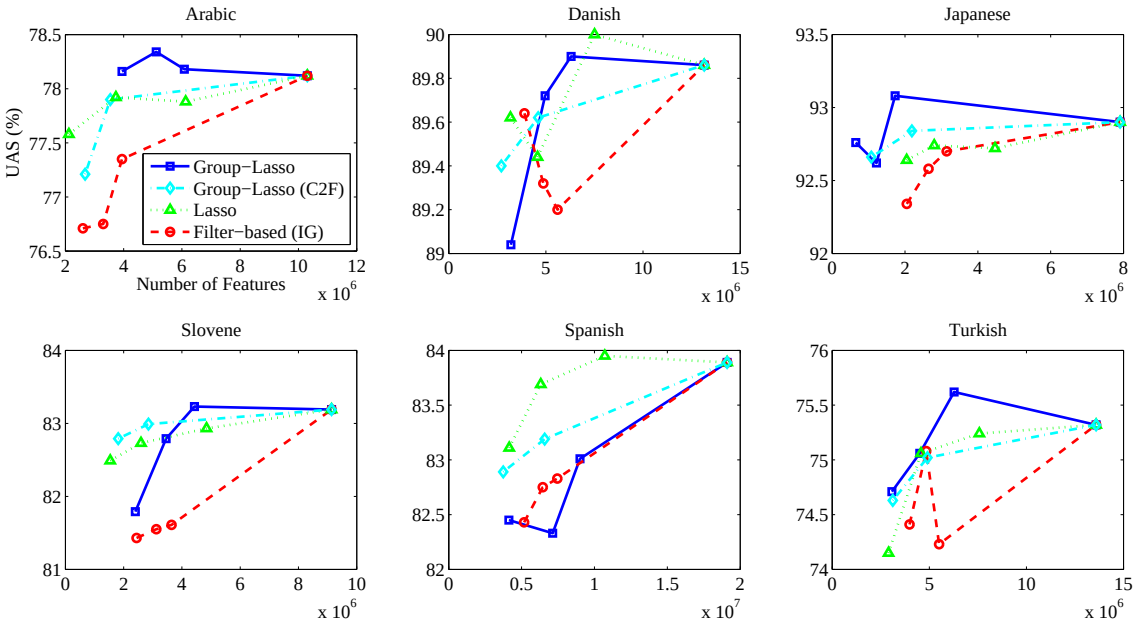


Figure 3: Comparison between non-overlapping group-Lasso, coarse-to-fine group-Lasso (C2F), and a filter-based method based on information gain for selecting feature templates in multilingual dependency parsing. The x-axis is the total number of features at different regularization levels, and the y-axis is the unlabeled attachment score. The plots illustrate how accurate the parsers are as a function of the model sparsity achieved, for each method. The standard Lasso (which does not select templates, but individual features) is also shown for comparison.

We use arc-factored models, for which exact inference is tractable (McDonald et al., 2005). We defined $M = 684$ feature templates for each candidate arc by conjoining the words, shapes, lemmas, and POS of the head and the modifier, as well as the contextual POS, and the distance and direction of attachment. We followed the same two-stage approach as before, and compared with a baseline which selects feature templates by ranking them according to the information gain criterion. This baseline assigns a score to each template T_m which reflects an empirical estimate of the mutual information between T_m and the binary variable A that indicates the presence/absence of a dependency link:

$$IG_m \triangleq \sum_{f \in T_m} \sum_{a \in \{0,1\}} P(f,a) \log_2 \frac{P(f,a)}{P(f)P(a)}, \quad (16)$$

where $P(f,a)$ is the joint probability of feature f firing and an arc being active ($a = 1$) or innactive ($a = 0$), and $P(f)$ and $P(a)$ are the corresponding marginals. All probabilities are estimated from the empirical counts of events observed in the data.

The results are plotted in Fig. 3, for budget sizes of 200, 300, and 400. We observe that for all but one language (Spanish is the exception), non-overlapping group-Lasso regularization is more effective at selecting feature templates than the information gain criterion, and slightly better than coarse-to-fine group-Lasso. For completeness, we also display the results obtained with a standard Lasso regularizer. Table 3 shows what kind of feature templates were most selected for each language. Some interesting patterns can be observed: morphologically-rich languages with small datasets (such as Turkish and Slovene) seem to avoid lexical features, arguably due to potential for overfitting; in Japanese, contextual POS appear to be specially relevant. It should be noted, however, that some of these patterns may be properties of the datasets rather than of the languages themselves.

6 Related Work

A variant of the online proximal gradient algorithm used in this paper was proposed by Martins et al.

	Ara.	Dan.	Jap.	Slo.	Spa.	Tur.
Bilexical	++	+			+	
Lex. → POS	+		+			
POS → Lex.	++	+	+		+	+
POS → POS			++	+		
Middle POS	++	++	++	++	++	++
Shape	++	++	++	++		
Direction		+	+	+	+	+
Distance	++	+	+	+	+	+

Table 3: Variation of feature templates that were selected accross languages. Each line groups together similar templates, involving lexical, contextual POS, word shape information, as well as attachment direction and length. Empty cells denote that very few or none of the templates in that category was selected; + denotes that some were selected; ++ denotes that most or all were selected.

(2011), along with a theoretical analysis. The focus there, however, was multiple kernel learning, hence overlapping groups were not considered in their experiments. Budget-driven shrinkage and the sparseptron are novel techniques, at the best of our knowledge. Apart from Martins et al. (2011), the only work we are aware of which combines structured sparsity with structured prediction is Schmidt and Murphy (2010); however, their goal is to predict the structure of graphical models, while we are mostly interested in the structure of the feature space. Schmidt and Murphy (2010) used to generative models, while our approach emphasizes discriminative learning.

Mixed norm regularization has been used for a while in statistics as a means to promote structured sparsity. Group Lasso is due to Bakin (1999) and Yuan and Lin (2006), after which a string of variants and algorithms appeared (Bach, 2008; Zhao et al., 2009; Jenatton et al., 2009; Friedman et al., 2010; Obozinski et al., 2010). The flat (non-overlapping) case has tight links with learning formalisms such as multiple kernel learning (Lanckriet et al., 2004) and multi-task learning (Caruana, 1997). The tree-structured case has been addressed by Kim and Xing (2010), Liu and Ye (2010) and Mairal et al. (2010), along with $L_{\infty,1}$ and $L_{2,1}$ regularization. Graph-structured groups are discussed in Jenatton et al. (2010), along with a DAG representation. In NLP, mixed norms have been used recently by Graça et al. (2009) in posterior regularization, and by Eisenstein et al. (2011) in a multi-task regression problem.

7 Conclusions

In this paper, we have explored two levels of structure in NLP problems: structure on the outputs, and structure on the feature space. We have shown how the latter can be useful in model design, through the use of regularizers which promote structured sparsity. We propose an online algorithm with minimal memory requirements for exploring large feature spaces. Our algorithm, which specializes into the *sparseptron*, yields a mechanism for selecting entire groups of features. We apply sparseptron for selecting feature templates in three structured prediction tasks, with advantages over filter-based methods, L_1, and L_2 regularization in terms of performance, compactness, and model interpretability.

Acknowledgments

We would like to thank all reviewers for their comments, Eric Xing for helpful discussions, and Slav Petrov for his comments on a draft version of this paper. A. M. was supported by a FCT/ICTI grant through the CMU-Portugal Program, and also by Priberam. This work was partially supported by the FET programme (EU FP7), under the SIMBAD project (contract 213250). N. S. was supported by NSF CAREER IIS-1054319.

References

Y. Altun, I. Tsochantaridis, and T. Hofmann. 2003. Hidden Markov support vector machines. In *Proc. of ICML*.

G. Andrew and J. Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proc. of ICML*. ACM.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. 2011. Convex optimization with sparsity-inducing norms. In *Optimization for Machine Learning*. MIT Press.

F. Bach. 2008. Exploring large feature spaces with hierarchical multiple kernel learning. *NIPS*, 21.

S. Bakin. 1999. *Adaptive regression and model selection in data mining problems*. Ph.D. thesis, Australian National University.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

B. Carpenter. 2008. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. Technical report, Technical report, Alias-i.

R. Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

S. F. Chen and R. Rosenfeld. 2000. A survey of smoothing techniques for maximum entropy models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

M. Collins. 2002a. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

M. Collins. 2002b. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proc. of ACL*.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online Passive-Aggressive Algorithms. *JMLR*, 7:551–585.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

J. Duchi and Y. Singer. 2009. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2873–2908.

J. Eisenstein, N. A. Smith, and E. P. Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proc. of ACL*.

M.A.T. Figueiredo. 2002. Adaptive sparseness using Jeffreys' prior. *Advances in Neural Information Processing Systems*.

J. Friedman, T. Hastie, and R. Tibshirani. 2010. A note on the group lasso and a sparse group lasso. Unpublished manuscript.

J. Gao, G. Andrew, M. Johnson, and K. Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proc. of ACL*.

J. Goodman. 2004. Exponential priors for maximum entropy models. In *Proc. of NAACL*.

J. Graça, K. Ganchev, B. Taskar, and F. Pereira. 2009. Posterior vs. parameter sparsity in latent variable models. *Advances in Neural Information Processing Systems*.

I. Guyon and A. Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

R. Jenatton, J.-Y. Audibert, and F. Bach. 2009. Structured variable selection with sparsity-inducing norms. Technical report, arXiv:0904.3523.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. 2010. Proximal methods for sparse hierarchical dictionary learning. In *Proc. of ICML*.

J. Kazama and J. Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proc. of EMNLP*.

S. Kim and E.P. Xing. 2010. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proc. of ICML*.

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

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. 2004. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72.

J. Langford, L. Li, and T. Zhang. 2009. Sparse online learning via truncated gradient. *JMLR*, 10:777–801.

T. Lavergne, O. Cappé, and F. Yvon. 2010. Practical very large scale CRFs. In *Proc. of ACL*.

J. Liu and J. Ye. 2010. Moreau-Yosida regularization for grouped tree structure learning. In *Advances in Neural Information Processing Systems*.

J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. 2010. Network flow algorithms for structured sparsity. In *Advances in Neural Information Processing Systems*.

A. F. T. Martins, N. A. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.

A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. 2011. Online learning of structured predictors with multiple kernels. In *Proc. of AISTATS*.

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

R. T. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

G. Obozinski, B. Taskar, and M.I. Jordan. 2010. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252.

S. Petrov and D. Klein. 2008a. Discriminative log-linear grammars with latent variables. *Advances in Neural Information Processing Systems*, 20:1153–1160.

S. Petrov and D. Klein. 2008b. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proc. of EMNLP*.

A. Quattoni, X. Carreras, M. Collins, and T. Darrell. 2009. An efficient projection for $l_{1,\infty}$ regularization. In *Proc. of ICML*.

E.F.T.K. Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*.

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

E.F.T.K. Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

M. Schmidt and K. Murphy. 2010. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proc. of AISTATS*.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML*.

N. Sokolovska, T. Lavergne, O. Cappé, and F. Yvon. 2010. Efficient learning of sparse conditional random fields for supervised sequence labelling. *IEEE Journal of Selected Topics in Signal Processing*, 4(6):953–964.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*.

R. Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B.*, pages 267–288.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *ICML*.

Y. Tsuruoka, J. Tsujii, and S. Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proc. of ACL*.

S.J. Wright, R. Nowak, and M.A.T. Figueiredo. 2009. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493.

L. Xiao. 2009. Dual averaging methods for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems*.

M. Yuan and Y. Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society (B)*, 68(1):49.

P. Zhao, G. Rocha, and B. Yu. 2009. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 37(6A):3468–3497.

H. Zou and T. Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 67(2):301–320.

Lexical Generalization in CCG Grammar Induction for Semantic Parsing

Tom Kwiatkowski[*] t.m.kwiatkowksi@sms.ed.ac.uk
Luke Zettlemoyer[†] lsz@cs.washington.edu
Sharon Goldwater[*] sgwater@inf.ed.ac.uk
Mark Steedman[*] steedman@inf.ed.ac.uk

[*]School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

[†]Computer Science & Engineering
University of Washington
Seattle, WA 98195

Abstract

We consider the problem of learning factored probabilistic CCG grammars for semantic parsing from data containing sentences paired with logical-form meaning representations. Traditional CCG lexicons list lexical items that pair words and phrases with syntactic and semantic content. Such lexicons can be inefficient when words appear repeatedly with closely related lexical content. In this paper, we introduce factored lexicons, which include both *lexemes* to model word meaning and *templates* to model systematic variation in word usage. We also present an algorithm for learning factored CCG lexicons, along with a probabilistic parse-selection model. Evaluations on benchmark datasets demonstrate that the approach learns highly accurate parsers, whose generalization performance benefits greatly from the lexical factoring.

1 Introduction

Semantic parsers automatically recover representations of meaning from natural language sentences. Recent work has focused on learning such parsers directly from corpora made up of sentences paired with logical meaning representations (Kate et al., 2005; Kate and Mooney, 2006; Wong and Mooney, 2006, 2007; Zettlemoyer and Collins, 2005, 2007; Lu et al., 2008; Kwiatkowski et al., 2010).

For example, in a flight booking domain we might have access to training examples such as:

Sentence: I want flights from Boston
Meaning: $\lambda x.flight(x) \wedge from(x, bos)$

and the goal is to learn a grammar that can map new, unseen, sentences onto their corresponding meanings, or logical forms.

One approach to this problem has developed algorithms for leaning probabilistic CCG grammars (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010). These grammars are well-suited to the task of semantic parsing, as they closely link syntax and semantics. They can be used to model a wide range of complex linguistic phenomena and are strongly lexicalized, storing all language-specific grammatical information directly with the words in the lexicon. For example, a typical learned lexicon might include entries such as:

(1) $flight \vdash N : \lambda x.flight(x)$
(2) $flight \vdash N/(S|NP) : \lambda f \lambda x.flight(x) \wedge f(x)$
(3) $flight \vdash N\backslash N : \lambda f \lambda x.flight(x) \wedge f(x)$
(4) $fare \vdash N : \lambda x.cost(x)$
(5) $fare \vdash N/(S|NP) : \lambda f \lambda x.cost(x) \wedge f(x)$
(6) $fare \vdash N\backslash N : \lambda f \lambda x.cost(x) \wedge f(x)$
(7) $Boston \vdash NP : bos$
(8) $Boston \vdash N\backslash N : \lambda f \lambda x.from(x, bos) \wedge f(x)$
(9) $New\ York \vdash NP : nyc$
(10) $New\ York \vdash N\backslash N : \lambda f \lambda x.from(x, nyc) \wedge f(x)$

Although lexicalization of this kind is useful for learning, as we will see, these grammars can also suffer from sparsity in the training data, since closely related entries must be repeatedly learned for all members of a certain class of words. For example, the list above shows a selection of lexical items that would have to be learned separately.

In this list, the word "flight" is paired with the predicate *flight* in three separate lexical items which are required for different syntactic contexts. Item

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1512–1523,
Edinburgh, Scotland, UK, July 27–31, 2011.

(1) has the standard *N* category for entries of this type, item (2) allows the use of the word "flight" with that-less relative clauses such as "flight departing Boston", and item (3) is useful for phrases with unconventional word order such as "from Boston flight to New York". Representing these three lexical items separately is inefficient, since each word of this class (such as "fare") will require three similarly structured lexical entries differing only in predicate name. There may also be systemtatic semantic variation between entries for a certain class of words. For example, in (6) "Boston" is paired with the constant *bos* that represents its meaning. However, item (7) also adds the predicate *from* to the logical form. This might be used to analyse somewhat elliptical, unedited sentences such as "Show me flights Boston to New York," which can be challenging for semantic parsers (Zettlemoyer and Collins, 2007).

This paper builds upon the insight that a large proportion of the variation between lexical items for a given class of words is systematic. Therefore it should be represented once and applied to a small set of basic lexical units. [1] We develop a *factored lexicon* that captures this insight by distinguishing *lexemes*, which pair words with logical constants, from *lexical templates*, which map lexemes to full lexical items. As we will see, this can lead to a significantly more compact lexicon that can be learned from less data. Each word or phrase will be associated with a few lexemes that can be combined with a shared set of general templates.

We develop an approach to learning factored, probabilistic CCG grammars for semantic parsing. Following previous work (Kwiatkowski et al., 2010), we make use of a higher-order unification learning scheme that defines a space of CCG grammars consistent with the (sentence, logical form) training pairs. However, instead of constructing fully specified lexical items for the learned grammar, we automatically generate sets of lexemes and lexical templates to model each example. This is a difficult learning problem, since the CCG analyses that are required to construct the final meaning representations are not explicitly labeled in the training data. Instead, we model them with hidden variables and develop an online learning approach that simultaneously estimates the parameters of a log-linear parsing model, while inducing the factored lexicon.

We evaluate the approach on the benchmark Atis and GeoQuery domains. This is a challenging setup, since the GeoQuery data has complex meaning representations and sentences in multiple languages, while the Atis data contains spontaneous, unedited text that can be difficult to analyze with a formal grammar representation. Our approach achieves at or near state-of-the-art recall across all conditions, despite having no English or domain-specific information built in. We believe that ours is the only system of sufficient generality to run with this degree of success on all of these datasets.

2 Related work

There has been significant previous work on learning semantic parsers from training sentences labelled with logical form meaning representations.

We extend a line of research that has addressed this problem by developing CCG grammar induction techniques. Zettlemoyer and Collins (2005, 2007) presented approaches that use hand generated, English-language specific rules to generate lexical items from logical forms as well as English specific type-shifting rules and relaxations of the CCG combinators to model spontaneous, unedited sentences. Zettlemoyer and Collins (2009) extends this work to the case of learning in context dependent environments. Kwiatkowski et al. (2010) described an approach for language-independent learning that replaces the hand-specified templates with a higher-order-unification-based lexical induction method, but their approach does not scale well to challenging, unedited sentences. The learning approach we develop for inducing factored lexicons is also language independent, but scales well to these challenging sentences.

There have been a number of other approaches for learning semantic parsers, including ones based on machine translation techniques (Papineni et al., 1997; Ramaswamy and Kleindienst, 2000; Wong and Mooney, 2006), parsing models (Miller et al., 1996; Ge and Mooney, 2006; Lu et al., 2008), in-

[1] A related tactic is commonly used in wide-coverage CCG parsers derived from treebanks, such as work by Hockenmaier and Steedman (2002) and Clark and Curran (2007). These parsers make extensive use of category-changing unary rules, to avoid data sparsity for systematically related categories (such as those related by type-raising). We will automatically learn to represent these types of generalizations in the factored lexicon.

ductive logic programming algorithms (Zelle and Mooney, 1996; Thompson and Mooney, 2002; Tang and Mooney, 2000), probabilistic automata (He and Young, 2005, 2006), and ideas from string kernels and support vector machines (Kate and Mooney, 2006; Nguyen et al., 2006).

More recent work has focused on training semantic parsers without supervision in the form of logical-form annotations. Clarke et al. (2010) and Liang et al. (2011) replace semantic annotations in the training set with target answers which are more easily available. Goldwasser et al. (2011) present work on unsupervised learning of logical form structure. However, all of these systems require significantly more domain and language specific initialization than the approach presented here.

Other work has learnt semantic analyses from text in the context of interactions in computational environments (Branavan et al. (2010), Vogel and Jurafsky (2010)); text grounded in partial observations of a world state (Liang et al., 2009); and from raw text alone (Poon and Domingos, 2009, 2010).

There is also related work that uses the CCG grammar formalism. Clark and Curran (2003) present a method for learning the parameters of a log-linear CCG parsing model from fully annotated normal–form parse trees. Watkinson and Manandhar (1999) describe an unsupervised approach for learning syntactic CCG lexicons. Bos et al. (2004) present an algorithm for building semantic representations from CCG parses but requires fully–specified CCG derivations in the training data.

3 Overview of the Approach

Here we give a formal definition of the problem and an overview of the learning approach.

Problem We will learn a semantic parser that takes a sentences x and returns a logical form z representing its underlying meaning. We assume we have input data $\{(x_i, z_i) | i = 1 \ldots n\}$ containing sentences x_i and logical forms z_i, for example $x_i =$"Show me flights to Boston" and $z_i = \lambda x.flight(x) \wedge to(x, bos)$.

Model We will represent the parser as a factored, probabilistic CCG (PCCG) grammar. A traditional CCG lexical item would fully specify the syntax and semantics for a word (reviewed in Section 4). For example, Boston $\vdash NP : bos$ represents the entry for the word "Boston" with syntactic category NP and meaning represented by the constant bos. Where a lexicon would usually list lexical items such as this, we instead use a factored lexicon (L, T) containing:

- A list of lexemes L. Each lexeme pairs a word or phrase with a list of logical constants that can be used to construct its meaning. For example, one lexeme might be $(\text{Boston}, [bos])$.
- A list of lexical templates T. Each template takes a lexeme and maps it on to a full lexical item. For example, there is a single template that can map the lexeme above to the final lexical entry Boston $\vdash NP : bos$.

We will make central use of this factored representation to provide a more compact representation of the lexicon that can be learned efficiently.

The factored PCCG will also contain a parameter vector, θ, that defines a log-linear distribution over the possible parses y, conditioned on the sentence x.

Learning Our approach for learning factored PCCGs extends the work of Kwiatkowski et al. (2010), as reviewed in Section 7. Specifically, we modify the lexical learning, to produce lexemes and templates, as well as the feature space of the model, but reuse the existing parameter estimation techniques and overall learning cycle, as described in Section 7.

We present the complete approach in three parts by describing the factored representation of the lexicon (Section 5), techniques for proposing potential new lexemes and templates (Section 6), and finally a complete learning algorithm (Section 7). However, the next section first reviews the required background on semantic parsing with CCG.

4 Background

4.1 Lambda Calculus

We represent the meanings of sentences, words and phrases with logical expressions that can contain constants, quantifiers, logical connectors and lambda abstractions. We construct the meanings of sentences from the meanings of words and phrases using lambda-calculus operations. We use a version of the typed lambda calculus (Carpenter, 1997), in which the basic types include e, for entities; t, for truth values; and i for numbers. We also have function types that are assigned to lambda expressions.

The expression $\lambda x.flight(x)$ takes an entity and returns a truth value, and has the function type $\langle e,t \rangle$.

4.2 Combinatory Categorial Grammar

CCG (Steedman, 1996, 2000) is a linguistic formalism that tightly couples syntax and semantics, and can be used to model a wide range of language phenomena. A traditional CCG grammar includes a lexicon Λ with entries like the following:

$$\begin{aligned} &flights \vdash N : \lambda x.flight(x) \\ &\quad to \vdash (N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y) \\ &Boston \vdash NP : bos \end{aligned}$$

where each lexical item w$\vdash X : h$ has words w, a syntactic category X, and a logical form h. For the first example, these are "flights," N, and $\lambda x.flight(x)$. In this paper, we introduce a new way of representing lexical items as (*lexeme*, *template*) pairs, as described in section 5.

CCG syntactic categories may be atomic (such as S or NP) or complex (such as $(N\backslash N)/NP$) where the slash combinators encode word order information. CCG uses a small set of *combinatory rules* to build syntactic parses and semantic representations *concurrently*. Two example combinatory rules are forward ($>$) and backward ($<$) *application*:

$$\begin{array}{llll} X/Y : f & Y : g & \Rightarrow & X : f(g) \qquad (>) \\ Y : g & X\backslash Y : f & \Rightarrow & X : f(g) \qquad (<) \end{array}$$

These rules apply to build syntactic and semantic derivations under the control of the word order information encoded in the slash directions of the lexical entries. For example, given the lexicon above, the phrase "flights to Boston" can be parsed to produce:

$$\dfrac{\dfrac{\text{flights}}{\begin{array}{c}N \\ \lambda x.flight(x)\end{array}} \quad \dfrac{\dfrac{\text{to}}{\begin{array}{c}(N\backslash N)/NP \\ \lambda y\lambda f\lambda x.f(x) \wedge to(x,y)\end{array}} \quad \dfrac{\text{Boston}}{\begin{array}{c}NP \\ bos\end{array}}}{\begin{array}{c}(N\backslash N) \\ \lambda f\lambda x.f(x) \wedge to(x,bos)\end{array}}>}{\begin{array}{c}N \\ \lambda x.flight(x) \wedge to(x,bos)\end{array}}<$$

where each step in the parse is labeled with the combinatory rule ($->$ or $-<$) that was used.

CCG also includes combinatory rules of forward ($>$ **B**) and backward ($<$ **B**) *composition*:

$$\begin{array}{llll} X/Y : f & Y/Z : g & \Rightarrow X/Z : \lambda x.f(g(x)) & (>\mathbf{B}) \\ Y\backslash Z : g & X\backslash Y : f & \Rightarrow X\backslash Z : \lambda x.f(g(x)) & (<\mathbf{B}) \end{array}$$

These rules allow a relaxed notion of constituency which helps limit the number of distinct CCG lexical items required.

To the standard forward and backward slashes of CCG we also add a vertical slash for which the direction of application is underspecified. We shall see examples of this in Section 10.

4.3 Probabilistic CCGs

Due to ambiguity in both the CCG lexicon and the order in which combinators are applied, there will be many parses for each sentence. We discriminate between competing parses using a log-linear model which has a feature vector ϕ and a parameter vector θ. The probability of a parse y that returns logical form z, given a sentence x is defined as:

$$P(y,z|x;\theta,\Lambda) = \frac{e^{\theta \cdot \phi(x,y,z)}}{\sum_{(y',z')} e^{\theta \cdot \phi(x,y',z')}} \qquad (1)$$

Section 8 fully defines the set of features used in the system presented. The most important of these control the generation of lexical items from (lexeme, template) pairs. Each (lexeme, template) pair used in a parse fires three features as we will see in more detail later.

The parsing, or inference, problem done at test time requires us to find the most likely logical form z given a sentence x, assuming the parameters θ and lexicon Λ are known:

$$f(x) = \arg\max_z p(z|x;\theta,\Lambda) \qquad (2)$$

where the probability of the logical form is found by summing over all parses that produce it:

$$p(z|x;\theta,\Lambda) = \sum_y p(y,z|x;\theta,\Lambda) \qquad (3)$$

In this approach the distribution over parse trees y is modeled as a hidden variable. The sum over parses in Eq. 3 can be calculated efficiently using the inside-outside algorithm with a CKY-style parsing algorithm.

To estimate the parameters themselves, we use stochastic gradient updates (LeCun et al., 1998). Given a set of n sentence-meaning pairs $\{(x_i,z_i) : i = 1...n\}$, we update the parameters θ iteratively, for each example i, by following the local gradient of the conditional log-likelihood objective

$O_i = \log P(z_i|x_i;\theta,\Lambda)$. The local gradient of the individual parameter θ_j associated with feature ϕ_j and training instance (x_i, z_i) is given by:

$$\frac{\partial O_i}{\partial \theta_j} = E_{p(y|x_i,z_i;\theta,\Lambda)}[\phi_j(x_i,y,z_i)] - E_{p(y,z|x_i;\theta,\Lambda)}[\phi_j(x_i,y,z)] \quad (4)$$

As with Eq. 3, all of the expectations in Eq. 4 are calculated through the use of the inside-outside algorithm on a pruned parse chart. For a sentence of length m, each parse chart span is pruned using a beam width proportional to $m^{\frac{2}{3}}$, to allow larger beams for shorter sentences.

5 Factored Lexicons

A factored lexicon includes a set L of lexemes and a set T of lexical templates. In this section, we formally define these sets, and describe how they are used to build CCG parses. We will use a set of lexical items from our running example to discuss the details of how the following lexical items:

(1) $flight \vdash N : \lambda x.flight(x)$
(2) $flight \vdash N/(S|NP) : \lambda f \lambda x.flight(x) \wedge f(x)$
...
(6) $Boston \vdash NP : bos$
(7) $Boston \vdash N\backslash N : \lambda f \lambda x.from(x,bos) \wedge f(x)$

are constructed from specific lexemes and templates.

5.1 Lexemes

A lexeme $(\mathrm{w}, \vec{c})$ pairs a word sequence w with an ordered list of logical constants $\vec{c} = [c_1 \ldots c_m]$. For example, item (1) and (2) above would come from a single lexeme $(\text{flight}, [flight])$. Similar lexemes would be represented for other predicates, for example $(\text{fare}, [cost])$. Lexemes also can contain multiple constants, for example $(\text{cheapest}, [argmin, cost])$, which we will see more examples of later.

5.2 Lexical Templates

A lexical template takes a lexeme and produces a lexical item. Templates have the general form

$$\lambda(\omega, \vec{v}).[\omega \vdash X : h_{\vec{v}}]$$

where $h_{\vec{v}}$ is a logical expression that contains variables from the list $\vec{v}$. Applying this template to the input lexeme $(\mathrm{w}, \vec{c})$ gives the full lexical item $\mathrm{w} \vdash X : h$ where the variable ω has been replaced with the wordspan w and the logical form h has been created by replacing each of the variables in $\vec{v}$ with the counterpart constant from $\vec{c}$. For example, the lexical item (6) above would be constructed from the lexeme $(\text{Boston}, [bos])$ using the template $\lambda(\omega, \vec{v}).[\omega \vdash NP : v_1]$. Items (1) and (2) would both be constructed from the single lexeme $(\text{flight}, [flight])$ with the two different templates $\lambda(\omega, \vec{v}).[\omega \vdash N : \lambda x.v_1(x)]$ and $\lambda(\omega, \vec{v}).[\omega \vdash N/(S|NP) : \lambda f \lambda x.v_1(x) \wedge f(x)]$

5.3 Parsing with a Factored Lexicon

In general, there can by many different (lexeme, template) pairs that produce the same lexical item. For example, lexical item (7) in our running example above can be constructed from the lexemes $(\text{Boston}, [bos])$ and $(\text{Boston}, [from, bos])$, given appropriate templates.

To model this ambiguity, we include the selection of a (lexeme, template) pair as a decision to be made while constructing a CCG parse tree. Given the lexical item produced by the chosen lexeme and template, parsing continues with the traditional combinators, as reviewed in Section 4.2. This direct integration allows for features that signal which lexemes and templates have been used while also allowing for well defined marginal probabilities, by summing over all ways of deriving a specific lexical item.

6 Learning Factored Lexicons

To induce factored lexicons, we will make use of two procedures, presented in this section, that factor lexical items into lexemes and templates. Section 7 will describe how this factoring operation is integrated into the complete learning algorithm.

6.1 Maximal Factorings

Given a lexical item l of the form $w \vdash X : h$ with words w, a syntactic category X, and a logical form h, we define the *maximal factoring* to be the unique (lexeme, template) pair that can be used to reconstruct l and includes all of the constants of h in the lexeme (listed in a fixed order based on an ordered traversal of h). For example, the maximal factoring for the lexical item $Boston \vdash NP : bos$ is the pair we saw before: $(\text{Boston}, [bos])$ and $\lambda(\omega, \vec{v}).[\omega \vdash NP : v_1]$. Similarly, the lexical item $Boston \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge from(x,bos)$ would be factored to produce $(\text{Boston}, [from, bos])$ and $\lambda(\omega, \vec{v}).[\omega \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge v_1(x, v_2)]$.

As we will see in Section 7, this notion of factor-

ing can be directly incorporated into existing algorithms that learn CCG lexicons. When the original algorithm would have added an entry l to the lexicon, we can instead compute the factoring of l and add the corresponding lexeme and template to the factored lexicon.

6.2 Introducing Templates with Content

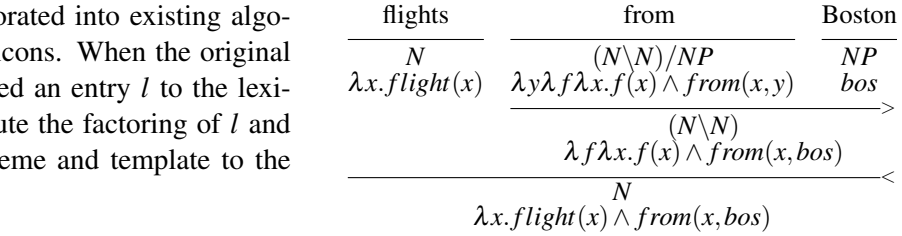
Maximal factorings, as just described, provide for significant lexical generalization but do not handle all of the cases needed to learn effectively. For instance, the maximal split for the item $Boston \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge from(x,bos)$ would introduce the lexeme $(\text{Boston}, [from, bos])$, which is suboptimal since each possible city would need a lexeme of this type, with the additional *from* constant included. Instead, we would ideally like to learn the lexeme $(\text{Boston}, [bos])$ and have a template that introduces the *from* constant. This would model the desired generalization with a single lexeme per city.

In order to permit the introduction of extra constants into lexical items, we allow the creation of templates that contain logical constants through *partial factorings*. For instance, the template below can introduce the predicate *from*

$$\lambda(\omega, \vec{v}).[\omega \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge from(x, v_1)]$$

The use of templates to introduce extra semantic constants into a lexical item is similar to, but more general than, the English-specific *type-shifting* rules used in Zettlemoyer and Collins (2007), which were introduced to model spontaneous, unedited text. They are useful, as we will see, in learning to recover semantic content that is implied, but not explicitly stated, such as our original motivating phrase "flights Boston to New York."

To propose templates which introduce semantic content, during learning, we build on the intuition that we need to recover from missing words, such as in the example above. In this scenario, there should also be other sentences that actually include the word, in our example this would be something like "flights from Boston." We will also assume that we have learned a good factored lexicon for the complete example that could produce the parse:



Given analyses of this form, we introduce new templates that will allow us to recover from missing words, for example if "from" was dropped. We identify commonly occurring nodes in the best parse trees found during training, in this case the nonterminal spanning "from Boston," and introduce templates that can produce the nonterminal, even if one of the words is missing. Here, this approach would introduce the desired template $\lambda(\omega, \vec{v}).[\omega \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge from(x, v_1)]$ for mapping the lexeme $(\text{Boston}, [bos])$ directly to the intermediate structure.

Not all templates introduced this way will model valid generalizations. However, we will incorporate them into a learning algorithm with indicator features that can be weighted to control their use. The next section presents the complete approach.

7 Learning Factored PCCGs

Our Factored Unification Based Learning (FUBL) method extends the UBL algorithm (Kwiatkowski et al., 2010) to induce factored lexicons, while also simultanously estimating the parameters of a log-linear CCG parsing model. In this section, we first review the NEW-LEX lexical induction procedure from UBL, and then present the FUBL algorithm.

7.1 Background: NEW-LEX

NEW-LEX generates lexical items by *splitting* and *merging* nodes in the best parse tree of each training example. Each parse node has a CCG category $X : h$ and a sequence of words w that it spans. We will present an overview of the approach using the running example with the phrase $w =$"in Boston" and the category $X : h = S\backslash NP : \lambda x.loc(x, bos)$, which is of the type commonly seen during learning. The splitting procedure is a two step process that first splits the logical form h, then splits the CCG syntactic category X and finally splits the string w.

The first step enumerates all possible splits of the logical form h into a pair of new expressions

(f, g) that can be used to reconstruct h by either function application ($h = f(g)$) or composition ($h = \lambda x.f(g(x))$). For example, one possible split is:

$$(f = \lambda y.\lambda x.loc(x,y)\ ,\ g = bos)$$

which corresponds to the function application case.

The next two steps enumerate all ways of splitting the syntactic category X and words w to introduce two new lexical items which can be recombined with CCG combinators (application or composition) to recreate the original parse node $X : h$ spanning w. In our example, one possibility would be:

$$(\text{in} \vdash (S\backslash NP)/NP : \lambda y.\lambda x.loc(x,y)\ ,\ \text{Boston} \vdash NP : bos)$$

which could be recombined with the forward application combinator from Section 4.2.

To assign categories while splitting, the grammar used by NEW-LEX only uses two atomic syntactic categories S and NP. This allows NEW-LEX to make use of a direct mapping from semantic type to syntactic category when proposing syntactic categories. In this schema, the standard syntactic category N is replaced by the category $S|NP$ which matches the type $\langle e,t \rangle$ and uses the vertical slash introduced in Section 4.2. We will see categories such as this in the evaluation.

7.2 The FUBL Algorithm

Figure 1 shows the FUBL learning algorithm. We assume training data $\{(x_i, z_i) : i = 1 \ldots n\}$ where each example is a sentence x_i paired with a logical form z_i. The algorithm induces a factored PCCG, including the lexemes L, templates T, and parameters θ.

The algorithm is online, repeatedly performing both lexical expansion (Step 1) and a parameter update (Step 2) for each training example. The overall approach is closely related to the UBL algorithm (Kwiatkowski et al., 2010), but includes extensions for updating the factored lexicon, as motivated in Section 6.

Initialization The model is initialized with a factored lexicon as follows. MAX-FAC is a function that takes a lexical item l and returns the maximal factoring of it, that is the unique, maximal (lexeme, template) pair that can be combined to construct l, as described in Section 6.1. We apply MAX-FAC to each of the training examples (x_i, z_i), creating a single way of producing the desired meaning z_i from a lexeme containing all of the words in x_i. The lexemes and templates created in this way provide the initial factored lexicon.

Inputs: Training set $\{(x_i, z_i) : i = 1 \ldots n\}$ where each example is a sentence x_i paired with a logical form z_i. Set of entity name lexemes L_e. Number of iterations J. Learning rate parameter α_0 and cooling rate parameter c. Empty lexeme set L. Empty template set T.

Definitions: NEW-LEX(y) returns a set of new lexical items from a parse y as described in Section 7.1. MAX-FAC(l) generates a (lexeme, template) pair from a lexical item l. PART-FAC(y) generates a set of templates from parse y. Both of these are described in Section 7.2. The distributions $p(y|x,z;\theta,(L,T))$ and $p(y,z|x;\theta,(L,T))$ are defined by the log-linear model described in Section 4.3.

Initialization:

- For $i = 1 \ldots n$
  - $(\psi, \pi) = \text{MAX-FAC}(x_i \vdash S : z_i)$
  - $L = L \cup \psi\ ,\ T = T \cup \pi$
- Set $L = L \cup L_e$.
- Initialize θ using coocurrence statistics, as described in Section 8.

Algorithm:

For $t = 1 \ldots J, i = 1 \ldots n$:

Step 1: (Add Lexemes and Templates)

- Let $y^* = \arg\max_y p(y|x_i, z_i; \theta, (L,T))$
- For $l \in \text{NEW-LEX}(y^*)$
  - $(\psi, \pi) = \text{MAX-FAC}(l)$
  - $L = L \cup \psi\ ,\ T = T \cup \pi$
- $\Pi = \text{PART-FAC}(y^*)\ ,\ T = T \cup \Pi$

Step 2: (Update Parameters)

- Let $\gamma = \frac{\alpha_0}{1 + c \times k}$ where $k = i + t \times n$.
- Let $\Delta = E_{p(y|x_i,z_i;\theta,(L,T))}[\phi(x_i,y,z_i)] - E_{p(y,z|x_i;\theta,(L,T))}[\phi(x_i,y,z)]$
- Set $\theta = \theta + \gamma\Delta$

Output: Lexemes L, templates T, and parameters θ.

Figure 1: The FUBL learning algorithm.

Step 1 The first step of the learning algorithm in Figure 1 adds lexemes and templates to the factored model given by performing manipulations on the highest scoring correct parse y^* of the current training example (x_i, z_i). First the NEW-LEX procedure is run on y^* as described in Section 6.1 to

generate new lexical items. We then use the function MAX-FAC to create the maximal factorings of each of these new lexical items as described in Section 6 and these are added to the factored representation of the lexicon. New templates can also be introduced through partial factorings of internal parse nodes as described in Section 6.2. These templates are generated by using the function PART-FAC to abstract over the wordspan and a subset of the constants contained in the internal parse nodes of y^*. This step allows for templates that introduce new semantic content to model elliptical language, as described in Section 6.2.

Step 2 The second step does a stochastic gradient descent update on the parameters θ used in the parsing model. This update is described in Section 4.3

Discussion The FUBL algorithm makes use of a direct online approach, where lexemes and templates are introduced in place while analyzing specific sentences. In general, this will overgeneralize; not all ways of combining lexemes and templates will produce high quality lexical items. However, the overall approach includes features, presented in Section 8, that can be used to learn which ones are best in practice. The complete algorithm iterates between adding new lexical content and updating the parameters of the parsing model with each procedure guiding the other.

8 Experimental setup

Data Sets We evaluate on two benchmark semantic parsing datasets: GeoQuery, which is made up of natural language queries to a database of geographical information; and Atis, which contains natural language queries to a flight booking system. The Geo880 dataset has 880 (English-sentence, logical-form) pairs split into a training set of 600 pairs and a test set of 280. The Geo250 data is a subset of the Geo880 sentences that have been translated into Japanese, Spanish and Turkish as well as the original English. We follow the standard evaluation procedure for Geo250, using 10-fold cross validation experiments with the same splits of the data as Wong and Mooney (2007). The Atis dataset contains 5410 (sentence, logical-form) pairs split into a 4480 example training set, a 480 example development set and a 450 example test set.

Evaluation Metrics We report exact match *Recall* (percentage of sentences for which the correct logical-form was returned), *Precision* (percentage of returned logical-forms that are correct) and *F1* (harmonic mean of Precision and Recall). For Atis we also report partial match Recall (percentage of correct literals returned), Precision (percentage of returned literals that are correct) and F1, computed as described by Zettlemoyer and Collins (2007).

Features We introduce two types of features to discriminate between parses: *lexical features* and *logical-form features*.

Lexical features fire on the lexemes and templates used to build the lexical items used in a parse. For each (lexeme,template) pair used to create a lexical item we have indicator features ϕ_l for the lexeme used, ϕ_t for the template used, and $\phi_{(l,t)}$ for the pair that was used. We assign the features on lexical templates a weight of 0.1 to prevent them from swamping the far less frequent but equally informative lexeme features.

Logical-form features are computed on the lambda-calculus expression z returned at the root of the parse. Each time a predicate p in z takes an argument a with type $Ty(a)$ in position i, it triggers two binary indicator features: $\phi_{(p,a,i)}$ for the predicate-argument relation; and $\phi_{(p,Ty(a),i)}$ for the predicate argument-type relation. Boolean operator features look at predicates that occurr together in conjunctions and disjunctions. For each variable v_i that fills argument slot i in two conjoined predicates p_1 and p_2 we introduce a binary indicator feature $\phi_{conj(i,p_1,p_2)}$. We introduce similar features $\phi_{disj(i,p_1,p_2)}$ for variables v_i that are shared by predicates in a disjunction.

Initialization The weights for lexeme features are initialized according to coocurrance statistics between words and logical constants. These are estimated with the Giza++ (Och and Ney, 2003) implementation of IBM Model 1. The initial weights for templates are set by adding -0.1 for each slash in the syntactic category and -2 if the template contains logical constants. Features on lexeme-template pairs and all parse features are initialized to zero.

Systems We compare performance to all recently-published, directly-comparable results. For GeoQuery, this includes the ZC05, ZC07 (Zettlemoyer

System	Exact Match		
	Rec.	Pre.	F1
ZC07	74.4	**87.3**	80.4
UBL	65.6	67.1	66.3
FUBL	**81.9**	82.1	**82.0**

Table 1: Performance on the Atis development set.

System	Exact Match			Partial Match		
	Rec.	Pre.	F1.	Rec.	Pre.	F1
ZC07	**84.6**	**85.8**	**85.2**	**96.7**	**95.1**	**95.9**
HY06	-	-	-	-	-	90.3
UBL	71.4	72.1	71.7	78.2	98.2	87.1
FUBL	82.8	82.8	82.8	95.2	93.6	94.6

Table 2: Performance on the Atis test set.

and Collins, 2005, 2007), λ-WASP (Wong and Mooney, 2007), UBL (Kwiatkowski et al., 2010) systems and DCS (Liang et al., 2011). For Atis, we report results from HY06 (He and Young, 2006), ZC07, and UBL.

9 Results

Tables 1-4 present the results on the Atis and Geoquery domains. In all cases, FUBL achieves at or near state-of-the-art recall (overall number of correct parses) when compared to directly comparable systems and it significantly outperforms UBL on Atis.

On Geo880 the only higher recall is achieved by DCS with prototypes - which uses significant English-specific resources, including manually specified lexical content, but does not require training sentences annotated with logical-forms. On Geo250, FUBL achieves the highest recall across languages. Each individual result should be interpreted with care, as a single percentage point corresponds to 2-3 sentences, but the overall trend is encouraging.

On the Atis development set, FUBL outperforms ZC07 by 7.5% of recall but on the Atis test set FUBL lags ZC07 by 2%. The reasons for this discrepancy are not clear, however, it is possible that the syntactic constructions found in the Atis test set do not exhibit the same degree of variation as those seen in the development set. This would negate the need for the very general lexicon learnt by FUBL.

Across the evaluations, despite achieving high recall, FUBL achieves significantly lower precision than ZC07 and λ-WASP. This illustrates the trade-off from having a very general model of proposing lexical structure. With the ability to skip unseen words, FUBL returns a parse for all of the Atis test sentences, since the factored lexicons we are learning can produce a very large number of lexical items. These parses are, however, not always correct.

System	Rec.	Pre.	F1
Labelled Logical Forms			
ZC05	79.3	**96.3**	87.0
ZC07	86.1	91.6	**88.8**
UBL	87.9	88.5	88.2
FUBL	**88.6**	88.6	88.6
Labelled Question Answers			
DCS	**91.1**	-	-

Table 3: Exact match accuracy on the Geo880 test set.

System	English			Spanish		
	Rec.	Pre.	F1	Rec.	Pre.	F1
λ-WASP	75.6	**91.8**	82.9	80.0	**92.5**	**85.8**
UBL	81.8	83.5	82.6	81.4	83.4	82.4
FUBL	**83.7**	83.7	**83.7**	**85.6**	85.8	85.7
System	Japanese			Turkish		
	Rec.	Pre.	F1	Rec.	Pre.	F1
λ-WASP	81.2	**90.1**	**85.8**	68.8	**90.4**	**78.1**
UBL	83.0	83.2	83.1	71.8	77.8	74.6
FUBL	**83.2**	83.8	83.5	**72.5**	73.7	73.1

Table 4: Exact-match accuracy on the Geo250 data set.

10 Analysis

The Atis results in Tables 1 and 2 highlight the advantages of factored lexicons. FUBL outperforms the UBL baseline by 16 and 11 points respectively in exact-match recall. Without making any modification to the CCG grammars or parsing combinators, we are able to induce a lexicon that is general enough model the natural occurring variations in the data, for example due to sloppy, unedited sentences.

Figure 2 shows a parse returned by FUBL for a sentence on which UBL failed. While the word "cheapest" is seen 208 times in the training data, in only a handful of these instances is it seen in the middle of an utterance. For this reason, UBL never proposes the lexical item, cheapest $\vdash NP\backslash(S|NP)/(S|NP) : \lambda f\lambda g.argmin(\lambda x.f(x) \wedge g(x), \lambda y.cost(y))$, which is used to parse the sentence in Figure 2. In contrast, FUBL uses a lexeme learned from the same word in different contexts, along with a template learnt from similar words in a similar context, to learn to per-

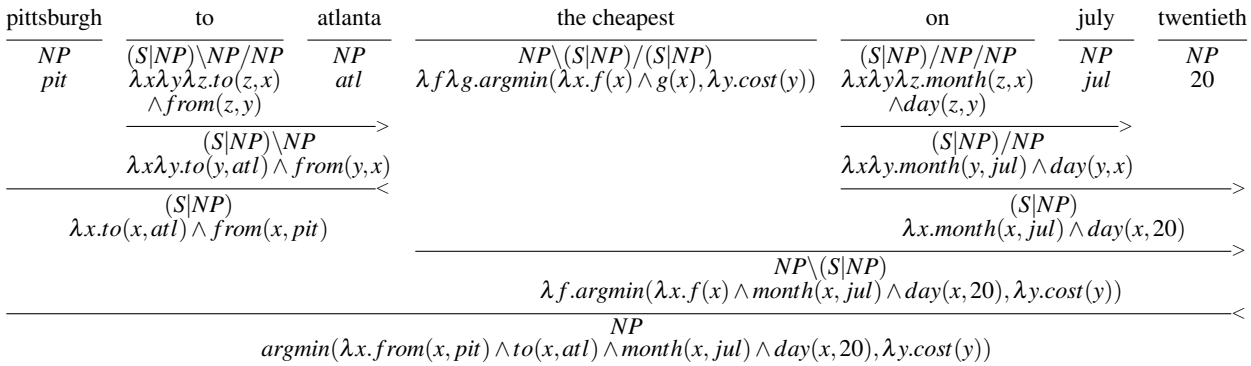


Figure 2: An example learned parse. FUBL can learn this type of analysis with novel combinations of lexemes and templates at test time, even if the individual words, like "cheapest," were never seen in similar syntactic constructions during training, as described in Section 10.

form the desired analysis.

As well as providing a new way to search the lexicon during training, the factored lexicon provides a way of proposing new, unseen, lexical items at test time. We find that new, non-*NP*, lexical items are used in 6% of the development set parses.

Interestingly, the addition of templates that introduce semantic content (as described in Section 6.2) account for only 1.2% of recall on the Atis development set. This is suprising as elliptical constructions are found in a much larger proportion of the sentences than this. In practice, FUBL learns to model many elliptical constructions with lexemes and templates introduced through maximal factorings. For example, the lexeme $(\text{to}, [from, to])$ can be used with the correct lexical template to deal with our motivating example "flights Boston to New York". Templates that introduce content are therefore only used in truly novel elliptical constructions for which an alternative analysis could not be learned.

Table 5 shows a selection of lexemes and templates learned for Atis. Examples 2 and 3 show that morphological variants of the same word must still be stored in separate lexemes. However, as these lexemes now share templates, the total number of lexical variants that must be learned is reduced.

Most common lexemes by type of constants in $\vec{c}$.			
1	e	(Boston, $[bos]$)	(Denver, $[den]$)
2	$\langle e,t \rangle$	(flight, $[flight]$)	(flights, $[flight]$)
3	$\langle e,i \rangle$	(fare, $[cost]$)	(fares, $[cost]$)
4	$\langle e, \langle e,t \rangle \rangle$	(from, $[from]$)	(to, $[to]$)
5	$\langle e,i \rangle, \langle e,t \rangle$	(cheapest, $[argmin, cost]$)	(earliest, $[argmin, dep\_time]$)
6	$\langle i, \langle i,t \rangle \rangle$, $\langle e,i \rangle$	(after, $[>, dep\_time]$)	(before, $[<, dep\_time]$)
Most common templates matching lexemes above.			
1	$\lambda(\omega, \vec{v}).\omega \vdash NP : v_1$		
2	$\lambda(\omega, \vec{v}).\omega \vdash S\|NP : \lambda x.v_1(x)$		
3	$\lambda(\omega, \vec{v}).\omega \vdash NP\|NP : \lambda x.v_1(x)$		
4	$\lambda(\omega, \vec{v}).\omega \vdash S\|NP/NP\backslash(S\|NP) : \lambda x\lambda y.v_1(x,y)$		
5	$\lambda(\omega, \vec{v}).\omega \vdash NP/(S\|NP) : \lambda f.v_1(\lambda x.f(x), \lambda y, v_2(y))$		
6	$\lambda(\omega, \vec{v}).\omega \vdash S\|NP\backslash(S\|NP)/NP : \lambda x\lambda y\lambda z.v_1(v_2(z),x) \wedge y(x)$		

Table 5: Example lexemes and templates learned from the Atis development set.

11 Discussion

We argued that factored CCG lexicons, which include both lexemes and lexical templates, provide a compact representation of lexical knowledge that can have advantages for learning. We also described a complete approach for inducing factored, probabilistic CCGs for semantic parsing, and demonstrated strong performance across a wider range of benchmark datasets that any previous approach.

In the future, it will also be important to explore morphological models, to better model variation within the existing lexemes. The factored lexical representation also has significant potential for lexical transfer learning, where we would need to learn new lexemes for each target application, but much of the information in the templates could, potentially, be ported across domains.

Acknowledgements

The work was supported in part by EU ERC Advanced Fellowship 249520 GRAMPLUS, and an ESPRC PhD studentship. We would like to thank Yoav Artzi for helpful discussions.

References

Bos, Johan, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the International Conference on Computational Linguistics*.

Branavan, S.R.K., Luke Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*.

Carpenter, Bob. 1997. *Type-Logical Semantics*. The MIT Press.

Clark, Stephen and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Clark, Stephen and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33(4):493–552.

Clarke, James, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*. Uppsala, Sweden, pages 18–27.

Ge, Ruifang and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

Goldwasser, Dan, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

He, Yulan and Steve Young. 2005. Semantic processing using the hidden vector state model. *Computer Speech and Language* .

He, Yulan and Steve Young. 2006. Spoken language understanding using the hidden vector state model. *Speech Communication* 48(3-4).

Hockenmaier, Julia and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*. Philadelphia, PA, pages 335–342.

Kate, Rohit J. and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Kate, Rohit J., Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the National Conference on Artificial Intelligence*.

Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Emperical Methods in Natural Language Processing*.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Liang, P., M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Liang, P., M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*.

Lu, Wei, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of The Conference on Empirical Methods in Natural Language Processing*.

Miller, Scott, David Stallard, Robert J. Bobrow, and Richard L. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proc. of the Association for Computational Linguistics*.

Nguyen, Le-Minh, Akira Shimazu, and Xuan-Hieu Phan. 2006. Semantic parsing with structured SVM ensemble classification models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

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

Papineni, K. A., S. Roukos, and T. R. Ward. 1997. Feature-based language understanding. In *Proceedings of European Conference on Speech Communication and Technology*.

Poon, Hoifung and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Poon, Hoifung and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Association for Computational Linguistics (ACL)*.

Ramaswamy, Ganesh N. and Jan Kleindienst. 2000. Hierarchical feature-based translation for scalable natural language understanding. In *Proceedings of International Conference on Spoken Language Processing*.

Steedman, Mark. 1996. *Surface Structure and Interpretation*. The MIT Press.

Steedman, Mark. 2000. *The Syntactic Process*. The MIT Press.

Tang, Lappoon R. and Raymond J. Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Thompson, Cynthia A. and Raymond J. Mooney. 2002. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research* 18.

Vogel, Adam and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Association for Computational Linguistics (ACL)*.

Watkinson, Stephen and Suresh Manandhar. 1999. Unsupervised lexical learning with categorial grammars using the LLL corpus. In *Proceedings of the 1st Workshop on Learning Language in Logic*.

Wong, Yuk Wah and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Wong, Yuk Wah and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Association for Computational Linguistics*.

Zelle, John M. and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, Luke S. and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zettlemoyer, Luke S. and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of The Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Named Entity Recognition in Tweets: An Experimental Study

Alan Ritter, Sam Clark, Mausam and Oren Etzioni
Computer Science and Engineering
University of Washington
Seattle, WA 98125, USA
{aritter,ssclark,mausam,etzioni}@cs.washington.edu

Abstract

People tweet more than 100 Million times daily, yielding a noisy, informal, but sometimes informative corpus of 140-character messages that mirrors the zeitgeist in an unprecedented manner. The performance of standard NLP tools is severely degraded on tweets. This paper addresses this issue by re-building the NLP pipeline beginning with part-of-speech tagging, through chunking, to named-entity recognition. Our novel T-NER system doubles F_1 score compared with the Stanford NER system. T-NER leverages the redundancy inherent in tweets to achieve this performance, using LabeledLDA to exploit Freebase dictionaries as a source of distant supervision. LabeledLDA outperforms co-training, increasing F_1 by 25% over ten common entity types.

Our NLP tools are available at: http://github.com/aritter/twitter_nlp

1 Introduction

Status Messages posted on Social Media websites such as Facebook and Twitter present a new and challenging style of text for language technology due to their noisy and informal nature. Like SMS (Kobus et al., 2008), tweets are particularly terse and difficult (See Table 1). Yet tweets provide a unique compilation of information that is more up-to-date and inclusive than news articles, due to the low-barrier to tweeting, and the proliferation of mobile devices.[1] The corpus of tweets already exceeds the size of the Library of Congress (Hachman, 2011) and is growing far more rapidly. Due to the volume of tweets, it is natural to consider named-entity recognition, information extraction, and text mining over tweets. Not surprisingly, the performance of "off the shelf" NLP tools, which were trained on news corpora, is weak on tweet corpora.

In response, we report on a re-trained "NLP pipeline" that leverages previously-tagged out-of-domain text, [2] tagged tweets, and unlabeled tweets to achieve more effective part-of-speech tagging, chunking, and named-entity recognition.

1	The Hobbit has FINALLY started filming! I cannot wait!
2	Yess! Yess! Its official Nintendo announced today that they Will release the Nintendo 3DS in north America march 27 for $250
3	Government confirms blast n nuclear plants n japan...don't knw wht s gona happen nw...

Table 1: Examples of noisy text in tweets.

We find that classifying named entities in tweets is a difficult task for two reasons. First, tweets contain a plethora of distinctive named entity types (Companies, Products, Bands, Movies, and more). Almost all these types (except for People and Locations) are relatively infrequent, so even a large sample of manually annotated tweets will contain few training examples. Secondly, due to Twitter's 140 character limit, tweets often lack sufficient context to determine an entity's type without the aid of background

[1] See the "trending topics" displayed on twitter.com

[2] Although tweets can be written on any subject, following convention we use the term "domain" to include text styles or genres such as Twitter, News or IRC Chat.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1524–1534,
Edinburgh, Scotland, UK, July 27–31, 2011.

knowledge.

To address these issues we propose a distantly supervised approach which applies LabeledLDA (Ramage et al., 2009) to leverage large amounts of unlabeled data in addition to large dictionaries of entities gathered from Freebase, and combines information about an entity's context across its mentions.

We make the following contributions:

1. We experimentally evaluate the performance of off-the-shelf news trained NLP tools when applied to Twitter. For example POS tagging accuracy drops from about 0.97 on news to 0.80 on tweets. By utilizing in-domain, out-of-domain, and unlabeled data we are able to substantially boost performance, for example obtaining a 52% increase in F_1 score on segmenting named entities.

2. We introduce a novel approach to *distant supervision* (Mintz et al., 2009) using Topic Models. LabeledLDA is applied, utilizing constraints based on an open-domain database (Freebase) as a source of supervision. This approach increases F_1 score by 25% relative to co-training (Blum and Mitchell, 1998; Yarowsky, 1995) on the task of classifying named entities in Tweets.

The rest of the paper is organized as follows. We successively build the NLP pipeline for Twitter feeds in Sections 2 and 3. We first present our approaches to shallow syntax – part of speech tagging (§2.1), and shallow parsing (§2.2). §2.3 describes a novel classifier that predicts the informativeness of capitalization in a tweet. All tools in §2 are used as features for named entity segmentation in §3.1. Next, we present our algorithms and evaluation for entity classification (§3.2). We describe related work in §4 and conclude in §5.

2 Shallow Syntax in Tweets

We first study two fundamental NLP tasks – POS tagging and noun-phrase chunking. We also discuss a novel capitalization classifier in §2.3. The outputs of all these classifiers are used in feature generation for named entity recognition in the next section.

For all experiments in this section we use a dataset of 800 randomly sampled tweets. All results (Tables 2, 4 and 5) represent 4-fold cross-validation experiments on the respective tasks.[3]

	Accuracy	Error Reduction
Majority Baseline (NN)	0.189	-
Word's Most Frequent Tag	0.760	-
Stanford POS Tagger	0.801	-
T-POS(PTB)	0.813	6%
T-POS(Twitter)	0.853	26%
T-POS(IRC + PTB)	0.869	34%
T-POS(IRC + Twitter)	0.870	35%
T-POS(PTB + Twitter)	0.873	36%
T-POS(PTB + IRC + Twitter)	0.883	41%

Table 2: POS tagging performance on tweets. By training on in-domain labeled data, in addition to annotated IRC chat data, we obtain a 41% reduction in error over the Stanford POS tagger.

2.1 Part of Speech Tagging

Part of speech tagging is applicable to a wide range of NLP tasks including named entity segmentation and information extraction.

Prior experiments have suggested that POS tagging has a very strong baseline: assign each word to its most frequent tag and assign each Out of Vocabulary (OOV) word the most common POS tag. This baseline obtained a 0.9 accuracy on the Brown corpus (Charniak et al., 1993). However, the application of a similar baseline on tweets (see Table 2) obtains a much weaker 0.76, exposing the challenging nature of Twitter data.

A key reason for this drop in accuracy is that Twitter contains far more OOV words than grammatical text. Many of these OOV words come from spelling variation, *e.g.*, the use of the word "n" for "in" in Table 1 example 3. Although NNP is the most frequent tag for OOV words, only about 1/3 are NNPs.

The performance of off-the-shelf news-trained POS taggers also suffers on Twitter data. The state-of-the-art Stanford POS tagger (Toutanova et al., 2003) improves on the baseline, obtaining an accuracy of 0.8. This performance is impressive given that its training data, the Penn Treebank WSJ (PTB), is so different in style from Twitter, however it is a huge drop from the 97% accuracy reported on the

[3]We used Brendan O'Connor's Twitter tokenizer

Gold	Predicted	Stanford Error	T-POS Error	Error Reduction
NN	NNP	0.102	0.072	29%
UH	NN	0.387	0.047	88%
VB	NN	0.071	0.032	55%
NNP	NN	0.130	0.125	4%
UH	NNP	0.200	0.036	82%

Table 3: Most common errors made by the Stanford POS Tagger on tweets. For each case we list the fraction of times the gold tag is misclassified as the predicted for both our system and the Stanford POS tagger. All verbs are collapsed into VB for compactness.

PTB. There are several reasons for this drop in performance. Table 3 lists common errors made by the Stanford tagger. First, due to unreliable capitalization, common nouns are often misclassified as proper nouns, and vice versa. Also, interjections and verbs are frequently misclassified as nouns. In addition to differences in vocabulary, the grammar of tweets is quite different from edited news text. For instance, tweets often start with a verb (where the subject 'I' is implied), as in: "watchng american dad."

To overcome these differences in style and vocabulary, we manually annotated a set of 800 tweets (16K tokens) with tags from the Penn TreeBank tag set for use as in-domain training data for our POS tagging system, T-POS.[4] We add new tags for the Twitter specific phenomena: retweets, @usernames, #hashtags, and urls. Note that words in these categories can be tagged with 100% accuracy using simple regular expressions. To ensure fair comparison in Table 2, we include a postprocessing step which tags these words appropriately for all systems.

To help address the issue of OOV words and lexical variations, we perform clustering to group together words which are distributionally similar (Brown et al., 1992; Turian et al., 2010). In particular, we perform hierarchical clustering using Jcluster (Goodman, 2001) on 52 million tweets; each word is uniquely represented by a bit string based on the path from the root of the resulting hierarchy to the word's leaf. We use the Brown clusters resulting from prefixes of 4, 8, and 12 bits. These clusters are often effective in capturing lexical variations, for example, following are lexical variations on the word "tomorrow" from one cluster after filtering out other words (most of which refer to days):

> '2m', '2ma', '2mar', '2mara', '2maro', '2marrow', '2mor', '2mora', '2moro', '2morow', '2morr', '2morro', '2morrow', '2moz', '2mr', '2mro', '2mrrw', '2mrw', '2mw', 'tmmrw', 'tmo', 'tmoro', 'tmorrow', 'tmoz', 'tmr', 'tmro', 'tmrow', 'tmrrow', 'tmrrw', 'tmrw', 'tmrww', 'tmw', 'tomaro', 'tomarow', 'tomarro', 'tomarrow', 'tomm', 'tommarow', 'tommarrow', 'tommoro', 'tommorow', 'tommorrow', 'tommorw', 'tommrow', 'tomo', 'tomolo', 'tomoro', 'tomorow', 'tomorro', 'tomorrw', 'tomoz', 'tomrw', 'tomz'

T-POS uses Conditional Random Fields[5] (Lafferty et al., 2001), both because of their ability to model strong dependencies between adjacent POS tags, and also to make use of highly correlated features (for example a word's identity in addition to prefixes and suffixes). Besides employing the Brown clusters computed above, we use a fairly standard set of features that include POS dictionaries, spelling and contextual features.

On a 4-fold cross validation over 800 tweets, T-POS outperforms the Stanford tagger, obtaining a 26% reduction in error. In addition we include 40K tokens of annotated IRC chat data (Forsythand and Martell, 2007), which is similar in style. Like Twitter, IRC data contains many misspelled/abbreviated words, and also more pronouns, and interjections, but fewer determiners than news. Finally, we also leverage 50K POS-labeled tokens from the Penn Treebank (Marcus et al., 1994).

Overall T-POS trained on 102K tokens (12K from Twitter, 40K from IRC and 50K from PTB) results in a 41% error reduction over the Stanford tagger, obtaining an accuracy of 0.883. Table 3 lists gains on some of the most common error types, for example, T-POS dramatically reduces error on interjections and verbs that are incorrectly classified as nouns by the Stanford tagger.

2.2 Shallow Parsing

Shallow parsing, or chunking is the task of identifying non-recursive phrases, such as noun phrases,

[4] Using MMAX2 (Müller and Strube, 2006) for annotation.

[5] We use MALLET (McCallum, 2002).

	Accuracy	Error Reduction
Majority Baseline (B-NP)	0.266	-
OpenNLP	0.839	-
T-CHUNK(CoNLL)	0.854	9%
T-CHUNK(Twitter)	0.867	17%
T-CHUNK(CoNLL + Twitter)	0.875	22%

Table 4: Token-Level accuracy at shallow parsing tweets. We compare against the OpenNLP chunker as a baseline.

verb phrases, and prepositional phrases in text. Accurate shallow parsing of tweets could benefit several applications such as Information Extraction and Named Entity Recognition.

Off the shelf shallow parsers perform noticeably worse on tweets, motivating us again to annotate in-domain training data. We annotate the same set of 800 tweets mentioned previously with tags from the CoNLL shared task (Tjong Kim Sang and Buchholz, 2000). We use the set of shallow parsing features described by Sha and Pereira (2003), in addition to the Brown clusters mentioned above. Part-of-speech tag features are extracted based on cross-validation output predicted by T-POS. For inference and learning, again we use Conditional Random Fields. We utilize 16K tokens of in-domain training data (using cross validation), in addition to 210K tokens of newswire text from the CoNLL dataset.

Table 4 reports T-CHUNK's performance at shallow parsing of tweets. We compare against the off-the shelf OpenNLP chunker[6], obtaining a 22% reduction in error.

2.3 Capitalization

A key orthographic feature for recognizing named entities is capitalization (Florian, 2002; Downey et al., 2007). Unfortunately in tweets, capitalization is much less reliable than in edited texts. In addition, there is a wide variety in the styles of capitalization. In some tweets capitalization is informative, whereas in other cases, non-entity words are capitalized simply for emphasis. Some tweets contain all lowercase words (8%), whereas others are in ALL CAPS (0.6%).

To address this issue, it is helpful to incorporate information based on the entire content of the message to determine whether or not its capitalization is informative. To this end, we build a capitalization classifier, T-CAP, which predicts whether or not a tweet is informatively capitalized. Its output is used as a feature for Named Entity Recognition. We manually labeled our 800 tweet corpus as having either "informative" or "uninformative" capitalization. The criteria we use for labeling is as follows: if a tweet contains any non-entity words which are capitalized, but do not begin a sentence, or it contains any entities which are not capitalized, then its capitalization is "uninformative", otherwise it is "informative".

	P	R	F_1
Majority Baseline	0.70	1.00	0.82
T-CAP	0.77	0.98	0.86

Table 5: Performance at predicting reliable capitalization.

For learning , we use Support Vector Machines.[7] The features used include: the fraction of words in the tweet which are capitalized, the fraction which appear in a dictionary of frequently lowercase/capitalized words but are not lowercase/capitalized in the tweet, the number of times the word 'I' appears lowercase and whether or not the first word in the tweet is capitalized. Results comparing against the majority baseline, which predicts capitalization is always informative, are shown in Table 5. Additionally, in §3 we show that features based on our capitalization classifier improve performance at named entity segmentation.

3 Named Entity Recognition

We now discuss our approach to named entity recognition on Twitter data. As with POS tagging and shallow parsing, off the shelf named-entity recognizers perform poorly on tweets. For example, applying the Stanford Named Entity Recognizer to one of the examples from Table 1 results in the following output:

> [Yess]$_{ORG}$! [Yess]$_{ORG}$! Its official [Nintendo]$_{LOC}$ announced today that they Will release the [Nintendo]$_{ORG}$ 3DS in north [America]$_{LOC}$ march 27 for $250

[6] http://incubator.apache.org/opennlp/

[7] http://www.chasen.org/~taku/software/TinySVM/

The OOV word 'Yess' is mistaken as a named entity. In addition, although the first occurrence of 'Nintendo' is correctly segmented, it is misclassified, whereas the second occurrence is improperly segmented – it should be the product "Nintendo 3DS". Finally "north America" should be segmented as a *LOCATION*, rather than just 'America'. In general, news-trained Named Entity Recognizers seem to rely heavily on capitalization, which we know to be unreliable in tweets.

Following Collins and Singer (1999), Downey et al. (2007) and Elsner et al. (2009), we treat classification and segmentation of named entities as separate tasks. This allows us to more easily apply techniques better suited towards each task. For example, we are able to use discriminative methods for named entity segmentation and distantly supervised approaches for classification. While it might be beneficial to jointly model segmentation and (distantly supervised) classification using a joint sequence labeling and topic model similar to that proposed by Sauper et al. (2010), we leave this for potential future work.

Because most words found in tweets are not part of an entity, we need a larger annotated dataset to effectively learn a model of named entities. We therefore use a randomly sampled set of 2,400 tweets for NER. All experiments (Tables 6, 8-10) report results using 4-fold cross validation.

3.1 Segmenting Named Entities

Because capitalization in Twitter is less informative than news, in-domain data is needed to train models which rely less heavily on capitalization, and also are able to utilize features provided by T-CAP.

We exhaustively annotated our set of 2,400 tweets (34K tokens) with named entities.[8] A convention on Twitter is to refer to other users using the @ symbol followed by their unique username. We deliberately choose not to annotate @usernames as entities in our data set because they are both unambiguous, and trivial to identify with 100% accuracy using a simple regular expression, and would only serve to inflate our performance statistics. While there is ambiguity as to the type of @usernames (for example, they can refer to people or companies), we believe they could be more easily classified using features of their associated user's profile than contextual features of the text.

[8]We found that including out-of-domain training data from the MUC competitions lowered performance at this task.

	P	R	F_1	F_1 inc.
Stanford NER	0.62	0.35	0.44	-
T-SEG(None)	0.71	0.57	0.63	43%
T-SEG(T-POS)	0.70	0.60	0.65	48%
T-SEG(T-POS, T-CHUNK)	0.71	0.61	0.66	50%
T-SEG(All Features)	0.73	0.61	0.67	52%

Table 6: Performance at segmenting entities varying the features used. "None" removes POS, Chunk, and capitalization features. Overall we obtain a 52% improvement in F_1 score over the Stanford Named Entity Recognizer.

T-SEG models Named Entity Segmentation as a sequence-labeling task using IOB encoding for representing segmentations (each word either begins, is inside, or is outside of a named entity), and uses Conditional Random Fields for learning and inference. Again we include orthographic, contextual and dictionary features; our dictionaries included a set of type lists gathered from Freebase. In addition, we use the Brown clusters and outputs of T-POS, T-CHUNK and T-CAP in generating features.

We report results at segmenting named entities in Table 6. Compared with the state-of-the-art news-trained Stanford Named Entity Recognizer (Finkel et al., 2005), T-SEG obtains a 52% increase in F_1 score.

3.2 Classifying Named Entities

Because Twitter contains many distinctive, and infrequent entity types, gathering sufficient training data for named entity classification is a difficult task. In any random sample of tweets, many types will only occur a few times. Moreover, due to their terse nature, individual tweets often do not contain enough context to determine the type of the entities they contain. For example, consider following tweet:

KKTNY in 45min.....

without any prior knowledge, there is not enough context to determine what type of entity "KKTNY" refers to, however by exploiting redundancy in the data (Downey et al., 2010), we can determine it is likely a reference to a television show since it of-

ten co-occurs with words such as *watching* and *premieres* in other contexts.[9]

In order to handle the problem of many infrequent types, we leverage large lists of entities and their types gathered from an open-domain ontology (Freebase) as a source of distant supervision, allowing use of large amounts of unlabeled data in learning.

Freebase Baseline: Although Freebase has very broad coverage, simply looking up entities and their types is inadequate for classifying named entities in context (0.38 F-score, §3.2.1). For example, according to Freebase, the mention 'China' could refer to a country, a band, a person, or a film. This problem is very common: 35% of the entities in our data appear in more than one of our (mutually exclusive) Freebase dictionaries. Additionally, 30% of entities mentioned on Twitter do not appear in any Freebase dictionary, as they are either too new (for example a newly released videogame), or are misspelled or abbreviated (for example 'mbp' is often used to refer to the "mac book pro").

Distant Supervision with Topic Models: To model unlabeled entities and their possible types, we apply LabeledLDA (Ramage et al., 2009), constraining each entity's distribution over topics based on its set of possible types according to Freebase. In contrast to previous weakly supervised approaches to Named Entity Classification, for example the Co-Training and Naïve Bayes (EM) models of Collins and Singer (1999), LabeledLDA models each entity string as a mixture of types rather than using a single hidden variable to represent the type of each mention. This allows information about an entity's distribution over types to be shared across mentions, naturally handling ambiguous entity strings whose mentions could refer to different types.

Each entity string in our data is associated with a bag of words found within a context window around all of its mentions, and also within the entity itself. As in standard LDA (Blei et al., 2003), each bag of words is associated with a distribution over topics, Multinomial(θ_e), and each topic is associated with a distribution over words, Multinomial(β_t). In addition, there is a one-to-one mapping between topics and Freebase type dictionaries. These dictionaries constrain θ_e, the distribution over topics for each entity string, based on its set of possible types, $FB[e]$. For example, θ_{Amazon} could correspond to a distribution over two types: *COMPANY*, and *LOCATION*, whereas θ_{Apple} might represent a distribution over *COMPANY*, and *FOOD*. For entities which aren't found in any of the Freebase dictionaries, we leave their topic distributions θ_e unconstrained. Note that in absence of any constraints LabeledLDA reduces to standard LDA, and a fully unsupervised setting similar to that presented by Elsner et. al. (2009).

In detail, the generative process that models our data for Named Entity Classification is as follows:

for each type: $t = 1 \dots T$ **do**
  Generate β_t according to symmetric Dirichlet distribution Dir(η).
end for
for each entity string $e = 1 \dots |E|$ **do**
  Generate θ_e over $FB[e]$ according to Dirichlet distribution Dir($\alpha_{FB[e]}$).
  for each word position $i = 1 \dots N_e$ **do**
    Generate $z_{e,i}$ from Mult(θ_e).
    Generate the word $w_{e,i}$ from Mult($\beta_{z_{e,i}}$).
  end for
end for

To infer values for the hidden variables, we apply Collapsed Gibbs sampling (Griffiths and Steyvers, 2004), where parameters are integrated out, and the $z_{e,i}$s are sampled directly.

In making predictions, we found it beneficial to consider θ_e^{train} as a prior distribution over types for entities which were encountered during training. In practice this sharing of information across contexts is very beneficial as there is often insufficient evidence in an isolated tweet to determine an entity's type. For entities which weren't encountered during training, we instead use a prior based on the distribution of types across all entities. One approach to classifying entities in context is to assume that θ_e^{train} is fixed, and that all of the words inside the entity mention and context, $\mathbf{w}$, are drawn based on a single topic, z, that is they are all drawn from Multinomial(β_z). We can then compute the posterior distribution over types in closed form with a simple application of Bayes rule:

$$P(z|\mathbf{w}) \propto \prod_{w \in \mathbf{w}} P(w|z:\beta) P(z:\theta_e^{\text{train}})$$

During development, however, we found that rather than making these assumptions, using Gibbs Sam-

[9] Kourtney & Kim Take New York.

Type	Top 20 Entities not found in Freebase dictionaries
PRODUCT	nintendo ds lite, apple ipod, generation black, ipod nano, apple iphone, gb black, xperia, ipods, verizon media, mac app store, kde, hd video, nokia n8, ipads, iphone/ipod, galaxy tab, samsung galaxy, playstation portable, nintendo ds, vpn
TV-SHOW	pretty little, american skins, nof, order svu, greys, kktny, rhobh, parks & recreation, parks & rec, dawson 's creek, big fat gypsy weddings, big fat gypsy wedding, winter wipeout, jersey shores, idiot abroad, royle, jerseyshore, mr . sunshine, hawaii five-0, new jersey shore
FACILITY	voodoo lounge, grand ballroom, crash mansion, sullivan hall, memorial union, rogers arena, rockwood music hall, amway center, el mocambo, madison square, bridgestone arena, cat club, le poisson rouge, bryant park, mandalay bay, broadway bar, ritz carlton, mgm grand, olympia theatre, consol energy center

Table 7: Example type lists produced by LabeledLDA. No entities which are shown were found in Freebase; these are typically either too new to have been added, or are misspelled/abbreviated (for example rhobh="Real Housewives of Beverly Hills"). In a few cases there are segmentation errors.

pling to estimate the posterior distribution over types performs slightly better. In order to make predictions, for each entity we use an informative Dirichlet prior based on θ_e^{train} and perform 100 iterations of Gibbs Sampling holding the hidden topic variables in the training data fixed (Yao et al., 2009). Fewer iterations are needed than in training since the type-word distributions, β have already been inferred.

3.2.1 Classification Experiments

To evaluate T-CLASS's ability to classify entity mentions in context, we annotated the 2,400 tweets with 10 types which are both popular on Twitter, and have good coverage in Freebase: *PERSON*, *GEO-LOCATION*, *COMPANY*, *PRODUCT*, *FACILITY*, *TV-SHOW*, *MOVIE*, *SPORTSTEAM*, *BAND*, and *OTHER*. Note that these type annotations are only used for evaluation purposes, and not used during training T-CLASS, which relies only on distant supervision. In some cases, we combine multiple Freebase types to create a dictionary of entities representing a single type (for example the *COMPANY* dictionary contains Freebase types */business/consumer_company* and */business/brand*). Because our approach does not rely on any manually labeled examples, it is straightforward to extend it for a different sets of types based on the needs of downstream applications.

Training: To gather unlabeled data for inference, we run T-SEG, our entity segmenter (from §3.1), on 60M tweets, and keep the entities which appear 100 or more times. This results in a set of 23,651 distinct entity strings. For each entity string, we collect words occurring in a context window of 3 words from all mentions in our data, and use a vocabulary of the 100K most frequent words. We run Gibbs sampling for 1,000 iterations, using the last sample to estimate entity-type distributions θ_e, in addition to type-word distributions β_t. Table 7 displays the 20 entities (not found in Freebase) whose posterior distribution θ_e assigns highest probability to selected types.

Results: Table 8 presents the classification results of T-CLASS compared against a majority baseline which simply picks the most frequent class (*PERSON*), in addition to the Freebase baseline, which only makes predictions if an entity appears in exactly one dictionary (*i.e.*, appears unambiguous). T-CLASS also outperforms a simple supervised baseline which applies a MaxEnt classifier using 4-fold cross validation over the 1,450 entities which were annotated for testing. Additionally we compare against the co-training algorithm of Collins and Singer (1999) which also leverages unlabeled data and uses our Freebase type lists; for seed rules we use the "unambiguous" Freebase entities. Our results demonstrate that T-CLASS outperforms the baselines and achieves a 25% increase in F_1 score over co-training.

Tables 9 and 10 present a breakdown of F_1 scores by type, both collapsing types into the standard classes used in the MUC competitions (*PERSON*, *LOCATION*, *ORGANIZATION*), and using the 10 popular Twitter types described earlier.

Entity Strings *vs.* Entity Mentions: DL-Cotrain and LabeledLDA use two different representations for the unlabeled data during learning. LabeledLDA groups together words across all mentions of an en-

System	P	R	F_1
Majority Baseline	0.30	0.30	0.30
Freebase Baseline	0.85	0.24	0.38
Supervised Baseline	0.45	0.44	0.45
DL-Cotrain	0.54	0.51	0.53
LabeledLDA	0.72	0.60	0.66

Table 8: Named Entity Classification performance on the 10 types. Assumes segmentation is given as in (Collins and Singer, 1999), and (Elsner et al., 2009).

Type	LL	FB	CT	SP	N
PERSON	0.82	0.48	0.65	0.83	436
LOCATION	0.74	0.21	0.55	0.67	372
ORGANIZATION	0.66	0.52	0.55	0.31	319
overall	0.75	0.39	0.59	0.49	1127

Table 9: F_1 classification scores for the 3 MUC types *PERSON*, *LOCATION*, *ORGANIZATION*. Results are shown using LabeledLDA (LL), Freebase Baseline (FB), DL-Cotrain (CT) and Supervised Baseline (SP). N is the number of entities in the test set.

Type	LL	FB	CT	SP	N
PERSON	0.82	0.48	0.65	0.86	436
GEO-LOC	0.77	0.23	0.60	0.51	269
COMPANY	0.71	0.66	0.50	0.29	162
FACILITY	0.37	0.07	0.14	0.34	103
PRODUCT	0.53	0.34	0.40	0.07	91
BAND	0.44	0.40	0.42	0.01	54
SPORTSTEAM	0.53	0.11	0.27	0.06	51
MOVIE	0.54	0.65	0.54	0.05	34
TV-SHOW	0.59	0.31	0.43	0.01	31
OTHER	0.52	0.14	0.40	0.23	219
overall	0.66	0.38	0.53	0.45	1450

Table 10: F_1 scores for classification broken down by type for LabeledLDA (LL), Freebase Baseline (FB), DL-Cotrain (CT) and Supervised Baseline (SP). N is the number of entities in the test set.

	P	R	F_1
DL-Cotrain-entity	0.47	0.45	0.46
DL-Cotrain-mention	0.54	0.51	**0.53**
LabeledLDA-entity	0.73	0.60	**0.66**
LabeledLDA-mention	0.57	0.52	0.54

Table 11: Comparing LabeledLDA and DL-Cotrain grouping unlabeled data by entities vs. mentions.

System	P	R	F_1
COTRAIN-NER (10 types)	0.55	0.33	0.41
T-NER(10 types)	0.65	0.42	**0.51**
COTRAIN-NER (PLO)	0.57	0.42	0.49
T-NER(PLO)	0.73	0.49	**0.59**
Stanford NER (PLO)	0.30	0.27	0.29

Table 12: Performance at predicting both segmentation and classification. Systems labeled with PLO are evaluated on the 3 MUC types *PERSON*, *LOCATION*, *ORGANIZATION*.

tity string, and infers a distribution over its possible types, whereas DL-Cotrain considers the entity mentions separately as unlabeled examples and predicts a type independently for each. In order to ensure that the difference in performance between LabeledLDA and DL-Cotrain is not simply due to this difference in representation, we compare both DL-Cotrain and LabeledLDA using both unlabeled datasets (grouping words by all mentions vs. keeping mentions separate) in Table 11. As expected, DL-Cotrain performs poorly when the unlabeled examples group mentions; this makes sense, since Co-Training uses a discriminative learning algorithm, so when trained on entities and tested on individual mentions, the performance decreases. Additionally, LabeledLDA's performance is poorer when considering mentions as "documents". This is likely due to the fact that there isn't enough context to effectively learn topics when the "documents" are very short (typically fewer than 10 words).

End to End System: Finally we present the end to end performance on segmentation and classification (T-NER) in Table 12. We observe that T-NER again outperforms co-training. Moreover, comparing against the Stanford Named Entity Recognizer on the 3 MUC types, T-NER doubles F_1 score.

4 Related Work

There has been relatively little previous work on building NLP tools for Twitter or similar text styles. Locke and Martin (2009) train a classifier to recognize named entities based on annotated Twitter data, handling the types *PERSON*, *LOCATION*, and *ORGANIZATION*. Developed in parallel to our work, Liu et al. (2011) investigate NER on the same 3 types, in addition to *PRODUCT*s and present a semi-

supervised approach using k-nearest neighbor. Also developed in parallel, Gimpell et al. (2011) build a POS tagger for tweets using 20 coarse-grained tags. Benson et. al. (2011) present a system which extracts artists and venues associated with musical performances. Recent work (Han and Baldwin, 2011; Gouws et al., 2011) has proposed lexical normalization of tweets which may be useful as a preprocessing step for the upstream tasks like POS tagging and NER. In addition Finin et. al. (2010) investigate the use of Amazon's Mechanical Turk for annotating Named Entities in Twitter, Minkov et. al. (2005) investigate person name recognizers in email, and Singh et. al. (2010) apply a minimally supervised approach to extracting entities from text advertisements.

In contrast to previous work, we have demonstrated the utility of features based on Twitter-specific POS taggers and Shallow Parsers in segmenting Named Entities. In addition we take a distantly supervised approach to Named Entity Classification which exploits large dictionaries of entities gathered from Freebase, requires no manually annotated data, and as a result is able to handle a larger number of types than previous work. Although we found manually annotated data to be very beneficial for named entity segmentation, we were motivated to explore approaches that don't rely on manual labels for classification due to Twitter's wide range of named entity types. Additionally, unlike previous work on NER in informal text, our approach allows the sharing of information across an entity's mentions which is quite beneficial due to Twitter's terse nature.

Previous work on Semantic Bootstrapping has taken a weakly-supervised approach to classifying named entities based on large amounts of unlabeled text (Etzioni et al., 2005; Carlson et al., 2010; Kozareva and Hovy, 2010; Talukdar and Pereira, 2010; McIntosh, 2010). In contrast, rather than predicting which classes an entity belongs to (e.g. a multi-label classification task), LabeledLDA estimates a *distribution* over its types, which is then useful as a prior when classifying mentions in context.

In addition there has been been work on Skip-Chain CRFs (Sutton, 2004; Finkel et al., 2005) which enforce consistency when classifying multiple occurrences of an entity within a document. Using topic models (e.g. LabeledLDA) for classifying named entities has a similar effect, in that information about an entity's distribution of possible types is shared across its mentions.

5 Conclusions

We have demonstrated that existing tools for POS tagging, Chunking and Named Entity Recognition perform quite poorly when applied to Tweets. To address this challenge we have annotated tweets and built tools trained on unlabeled, in-domain and out-of-domain data, showing substantial improvement over their state-of-the art news-trained counterparts, for example, T-POS outperforms the Stanford POS Tagger, reducing error by 41%. Additionally we have shown the benefits of features generated from T-POS and T-CHUNK in segmenting Named Entities.

We identified named entity classification as a particularly challenging task on Twitter. Due to their terse nature, tweets often lack enough context to identify the types of the entities they contain. In addition, a plethora of distinctive named entity types are present, necessitating large amounts of training data. To address both these issues we have presented and evaluated a distantly supervised approach based on LabeledLDA, which obtains a 25% increase in F_1 score over the co-training approach to Named Entity Classification suggested by Collins and Singer (1999) when applied to Twitter.

Our POS tagger, Chunker Named Entity Recognizer are available for use by the research community: `http://github.com/aritter/twitter_nlp`

Acknowledgments

We would like to thank Stephen Soderland, Dan Weld and Luke Zettlemoyer, in addition to the anonymous reviewers for helpful comments on a previous draft. This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-11-1-0294, Navy STTR contract N00014-10-M-0304, a National Defense Science and Engineering Graduate (NDSEG) Fellowship 32 CFR 168a and carried out at the University of Washington's Turing Center.

References

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *The 49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon, USA. To appear.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*

Avrim Blum and Tom M. Mitchell. 1998. Combining labeled and unlabeled sata with co-training. In *COLT*, pages 92–100.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. In *AAAI*, pages 784–789.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Empirical Methods in Natural Language Processing*.

Doug Downey, Matthew Broadhead, and Oren Etzioni. 2007. Locating complex named entities in web text. In *Proceedings of the 20th international joint conference on Artifical intelligence*.

Doug Downey, Oren Etzioni, and Stephen Soderland. 2010. Analysis of a probabilistic model of redundancy in unsupervised information extraction. *Artif. Intell.*, 174(11):726–748.

Micha Elsner, Eugene Charniak, and Mark Johnson. 2009. Structured generative models for unsupervised named-entity clustering. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in Twitter data with crowdsourcing. In *Proceedings of the NAACL Workshop on Creating Speech and Text Language Data With Amazon's Mechanical Turk*. Association for Computational Linguistics, June.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05.

Radu Florian. 2002. Named entity recognition as a house of cards: classifier stacking. In *Proceedings of the 6th conference on Natural language learning - Volume 20*, COLING-02.

Eric N. Forsythand and Craig H. Martell. 2007. Lexical and discourse analysis of online chat dialog. In *Proceedings of the International Conference on Semantic Computing*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *ACL*.

Joshua T. Goodman. 2001. A bit of progress in language modeling. Technical report, Microsoft Research.

Stephan Gouws, Donald Metzler, Congxing Cai, and Eduard Hovy. 2011. Contextual bearing on linguistic variation in social media. In *ACL Workshop on Language in Social Media*, Portland, Oregon, USA. To appear.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, April.

Mark Hachman. 2011. Humanity's tweets: Just 20 terabytes. In *PCMAG.COM*.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *The 49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon, USA. To appear.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing sms: are two metaphors better than one ? In *COLING*, pages 441–448.

Zornitsa Kozareva and Eduard H. Hovy. 2010. Not all seeds are equal: Measuring the quality of text mining seeds. In *HLT-NAACL*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *ACL*.

Brian Locke and James Martin. 2009. Named entity recognition: Adapting to microblogging. In *Senior Thesis, University of Colorado*.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*.

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

Tara McIntosh. 2010. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10.

Einat Minkov, Richard C. Wang, and William W. Cohen. 2005. Extracting personal names from email: applying named entity recognition to informal text. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 443–450, Morristown, NJ, USA. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP 2009*.

Christoph Müller and Michael Strube. 2006. Multi-level annotation of linguistic data with MMAX2. In Sabine Braun, Kurt Kohn, and Joybrato Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled lda: a supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 248–256, Morristown, NJ, USA. Association for Computational Linguistics.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2010. Incorporating content structure into text analysis applications. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 377–387, Morristown, NJ, USA. Association for Computational Linguistics.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03.

Sameer Singh, Dustin Hillard, and Chris Leggetter. 2010. Minimally-supervised extraction of entities from text advertisements. In *Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*.

Charles Sutton. 2004. Collective segmentation and labeling of distant entities in information extraction.

Partha Pratim Talukdar and Fernando Pereira. 2010. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1473–1481. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7*, ConLL '00.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95.

Identifying Relations for Open Information Extraction

Anthony Fader, Stephen Soderland, and Oren Etzioni
University of Washington, Seattle
{afader,soderlan,etzioni}@cs.washington.edu

Abstract

Open Information Extraction (IE) is the task of extracting assertions from massive corpora without requiring a pre-specified vocabulary. This paper shows that the output of state-of-the-art Open IE systems is rife with uninformative and incoherent extractions. To overcome these problems, we introduce two simple syntactic and lexical constraints on binary relations expressed by verbs. We implemented the constraints in the REVERB Open IE system, which more than doubles the area under the precision-recall curve relative to previous extractors such as TEXTRUNNER and WOE^{pos}. More than 30% of REVERB's extractions are at precision 0.8 or higher—compared to virtually none for earlier systems. The paper concludes with a detailed analysis of REVERB's errors, suggesting directions for future work.[1]

1 Introduction and Motivation

Typically, Information Extraction (IE) systems learn an extractor for each target relation from labeled training examples (Kim and Moldovan, 1993; Riloff, 1996; Soderland, 1999). This approach to IE does not scale to corpora where the number of target relations is very large, or where the target relations cannot be specified in advance. Open IE solves this problem by identifying *relation phrases*—phrases that denote relations in English sentences (Banko et al., 2007). The automatic identification of relation phrases enables the extraction of *arbitrary* relations from sentences, obviating the restriction to a pre-specified vocabulary.

Open IE systems have achieved a notable measure of success on massive, open-domain corpora drawn from the Web, Wikipedia, and elsewhere. (Banko et al., 2007; Wu and Weld, 2010; Zhu et al., 2009). The output of Open IE systems has been used to support tasks like learning selectional preferences (Ritter et al., 2010), acquiring common sense knowledge (Lin et al., 2010), and recognizing entailment (Schoenmackers et al., 2010; Berant et al., 2011). In addition, Open IE extractions have been mapped onto existing ontologies (Soderland et al., 2010).

We have observed that two types of errors are frequent in the output of Open IE systems such as TEXTRUNNER and WOE: *incoherent extractions* and *uninformative extractions*.

Incoherent extractions are cases where the extracted relation phrase has no meaningful interpretation (see Table 1 for examples). Incoherent extractions arise because the learned extractor makes a sequence of decisions about whether to include each word in the relation phrase, often resulting in incomprehensible predictions. To solve this problem, we introduce a syntactic constraint: every multi-word relation phrase must begin with a verb, end with a preposition, and be a contiguous sequence of words in the sentence. Thus, the identification of a relation phrase is made in one fell swoop instead of on the basis of multiple, word-by-word decisions.

Uninformative extractions are extractions that omit critical information. For example, consider the sentence "Faust made a deal with the devil." Previ-

[1] The source code for REVERB is available at http://reverb.cs.washington.edu/

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1535–1545,
Edinburgh, Scotland, UK, July 27–31, 2011.

ous Open IE systems return the uninformative

(Faust, made, a deal)

instead of

(Faust, made a deal with, the devil).

This type of error is caused by improper handling of relation phrases that are expressed by a combination of a verb with a noun, such as light verb constructions (LVCs). An LVC is a multi-word expression composed of a verb and a noun, with the noun carrying the semantic content of the predicate (Grefenstette and Teufel, 1995; Stevenson et al., 2004; Allerton, 2002). Table 2 illustrates the wide range of relations expressed this way, which are not captured by existing open extractors. Our syntactic constraint leads the extractor to include nouns in the relation phrase, solving this problem.

Although the syntactic constraint significantly reduces incoherent and uninformative extractions, it allows overly-specific relation phrases such as *is offering only modest greenhouse gas reduction targets at*. To avoid overly-specific relation phrases, we introduce an intuitive lexical constraint: a binary relation phrase ought to appear with at least a minimal number of distinct argument pairs in a large corpus.

In summary, this paper articulates two simple but surprisingly powerful constraints on how binary relationships are expressed via verbs in English sentences, and implements them in the REVERB Open IE system. We release REVERB and the data used in our experiments to the research community.

The rest of the paper is organized as follows. Section 2 analyzes previous work. Section 3 defines our constraints precisely. Section 4 describes REVERB, our implementation of the constraints. Section 5 reports on our experimental results. Section 6 concludes with a summary and discussion of future work.

2 Previous Work

Open IE systems like TEXTRUNNER (Banko et al., 2007), WOE^{pos}, and WOE^{parse} (Wu and Weld, 2010) focus on extracting binary relations of the form (arg1, relation phrase, arg2) from text. These systems all use the following three-step method:

1. **Label:** Sentences are automatically labeled with extractions using heuristics or distant supervision.

Sentence	Incoherent Relation
The guide *contains* dead links and *omits* sites.	contains omits
The Mark 14 *was central* to the *torpedo* scandal of the fleet.	was central torpedo
They *recalled* that Nungesser *began* his career as a precinct leader.	recalled began

Table 1: Examples of incoherent extractions. Incoherent extractions make up approximately 13% of TEXTRUNNER's output, 15% of WOE^{pos}'s output, and 30% of WOE^{parse}'s output.

is	is an album by, is the author of, is a city in
has	has a population of, has a Ph.D. in, has a cameo in
made	made a deal with, made a promise to
took	took place in, took control over, took advantage of
gave	gave birth to, gave a talk at, gave new meaning to
got	got tickets to, got a deal on, got funding from

Table 2: Examples of uninformative relations (left) and their completions (right). Uninformative relations occur in approximately 4% of WOE^{parse}'s output, 6% of WOE^{pos}'s output, and 7% of TEXTRUNNER's output.

2. **Learn:** A relation phrase extractor is learned using a sequence-labeling graphical model (*e.g.*, CRF).

3. **Extract:** the system takes a sentence as input, identifies a candidate pair of NP arguments (arg1, arg2) from the sentence, and then uses the learned extractor to label each word between the two arguments as part of the relation phrase or not.

The extractor is applied to the successive sentences in the corpus, and the resulting extractions are collected.

This method faces several challenges. First, the training phase requires a large number of labeled training examples (*e.g.*, 200,000 heuristically-labeled sentences for TEXTRUNNER and 300,000 for WOE). Heuristic labeling of examples obviates hand labeling but results in noisy labels and distorts the distribution of examples. Second, the extraction step is posed as a sequence-labeling problem, where each word is assigned its own label. Because each assignment is uncertain, the likelihood that the extracted relation phrase is flawed increases with the length of the sequence. Finally, the extractor

chooses an extraction's arguments heuristically, and cannot backtrack over this choice. This is problematic when a word that belongs in the relation phrase is chosen as an argument (for example, *deal* from the "made a deal with" sentence).

Because of the feature sets utilized in previous work, the learned extractors ignore both "holistic" aspects of the relation phrase (*e.g.*, is it contiguous?) as well as lexical aspects (*e.g.*, how many instances of this relation are there?). Thus, as we show in Section 5, systems such as TEXTRUNNER are unable to learn the constraints embedded in REVERB. Of course, a learning system, utilizing a different hypothesis space, and an appropriate set of training examples, *could* potentially learn and refine the constraints in REVERB. This is a topic for future work, which we consider in Section 6.

The first Open IE system was TEXTRUNNER (Banko et al., 2007), which used a Naive Bayes model with unlexicalized POS and NP-chunk features, trained using examples heuristically generated from the Penn Treebank. Subsequent work showed that utilizing a linear-chain CRF (Banko and Etzioni, 2008) or Markov Logic Network (Zhu et al., 2009) can lead to improved extraction. The WOE systems introduced by Wu and Weld make use of Wikipedia as a source of training data for their extractors, which leads to further improvements over TEXTRUNNER (Wu and Weld, 2010). Wu and Weld also show that dependency parse features result in a dramatic increase in precision and recall over shallow linguistic features, but at the cost of extraction speed.

Other approaches to large-scale IE have included Preemptive IE (Shinyama and Sekine, 2006), On-Demand IE (Sekine, 2006), and weak supervision for IE (Mintz et al., 2009; Hoffmann et al., 2010). Preemptive IE and On-Demand IE avoid relation-specific extractors, but rely on document and entity clustering, which is too costly for Web-scale IE. Weakly supervised methods use an existing ontology to generate training data for learning relation-specific extractors. While this allows for learning relation-specific extractors at a larger scale than what was previously possible, the extractions are still restricted to a specific ontology.

Many systems have used syntactic patterns based on verbs to extract relation phrases, usually relying on a full dependency parse of the input sentence (Lin and Pantel, 2001; Stevenson, 2004; Specia and Motta, 2006; Kathrin Eichler and Neumann, 2008). Our work differs from these approaches by focusing on relation phrase patterns expressed in terms of POS tags and NP chunks, instead of full parse trees. Banko and Etzioni (Banko and Etzioni, 2008) showed that a small set of POS-tag patterns cover a large fraction of relationships in English, but never incorporated the patterns into an extractor. This paper reports on a substantially improved model of binary relation phrases, which increases the recall of the Banko-Etzioni model (see Section 3.3). Further, while previous work in Open IE has mainly focused on syntactic patterns for relation extraction, we introduce a lexical constraint that boosts precision and recall.

Finally, Open IE is closely related to semantic role labeling (SRL) (Punyakanok et al., 2008; Toutanova et al., 2008) in that both tasks extract relations and arguments from sentences. However, SRL systems traditionally rely on syntactic parsers, which makes them susceptible to parser errors and substantially slower than Open IE systems such as REVERB. This difference is particularly important when operating on the Web corpus due to its size and heterogeneity. Finally, SRL requires hand-constructed semantic resources like Propbank and Framenet (Martha and Palmer, 2002; Baker et al., 1998) as input. In contrast, Open IE systems require no relation-specific training data. ReVerb, in particular, relies on its explicit lexical and syntactic constraints, which have no correlate in SRL systems. For a more detailed comparison of SRL and Open IE, see (Christensen et al., 2010).

3 Constraints on Relation Phrases

In this section we introduce two constraints on relation phrases: a syntactic constraint and a lexical constraint.

3.1 Syntactic Constraint

The syntactic constraint serves two purposes. First, it eliminates incoherent extractions, and second, it reduces uninformative extractions by capturing relation phrases expressed by a verb-noun combination, including light verb constructions.

$$V \mid VP \mid VW^*P$$
V = verb particle? adv?
W = (noun | adj | adv | pron | det)
P = (prep | particle | inf. marker)

Figure 1: A simple part-of-speech-based regular expression reduces the number of incoherent extractions like *was central torpedo* and covers relations expressed via light verb constructions like *gave a talk at*.

The syntactic constraint requires the relation phrase to match the POS tag pattern shown in Figure 1. The pattern limits relation phrases to be either a verb (*e.g.*, *invented*), a verb followed immediately by a preposition (*e.g.*, *located in*), or a verb followed by nouns, adjectives, or adverbs ending in a preposition (*e.g.*, *has atomic weight of*). If there are multiple possible matches in a sentence for a single verb, the longest possible match is chosen. Finally, if the pattern matches multiple adjacent sequences, we merge them into a single relation phrase (*e.g.*, *wants to extend*). This refinement enables the model to readily handle relation phrases containing multiple verbs. A consequence of this pattern is that the relation phrase must be a contiguous span of words in the sentence.

The syntactic constraint eliminates the incoherent relation phrases returned by existing systems. For example, given the sentence

> Extendicare agreed to buy Arbor Health Care for about US $432 million in cash and assumed debt.

TEXTRUNNER returns the extraction

(Arbor Health Care, for assumed, debt).

The phrase *for assumed* is clearly not a valid relation phrase: it begins with a preposition and splices together two distant words in the sentence. The syntactic constraint prevents this type of error by simply restricting relation phrases to match the pattern in Figure 1.

The syntactic constraint reduces uninformative extractions by capturing relation phrases expressed via LVCs. For example, the POS pattern matched against the sentence "Faust made a deal with the Devil," would result in the relation phrase *made a deal with*, instead of the uninformative *made*.

Finally, we require the relation phrase to appear between its two arguments in the sentence. This is a common constraint that has been implicitly enforced in other open extractors.

3.2 Lexical Constraint

While the syntactic constraint greatly reduces uninformative extractions, it can sometimes match relation phrases that are so specific that they have only a few possible instances, even in a Web-scale corpus. Consider the sentence:

> The Obama administration is offering only modest greenhouse gas reduction targets at the conference.

The POS pattern will match the phrase:

is offering only modest greenhouse gas reduction targets at (1)

Thus, there are phrases that satisfy the syntactic constraint, but are not relational.

To overcome this limitation, we introduce a lexical constraint that is used to separate valid relation phrases from overspecified relation phrases, like the example in (1). The constraint is based on the intuition that a valid relation phrase should take many distinct arguments in a large corpus. The phrase in (1) is specific to the argument pair *(Obama administration, conference)*, so it is unlikely to represent a *bona fide* relation. We describe the implementation details of the lexical constraint in Section 4.

3.3 Limitations

Our constraints represent an idealized model of relation phrases in English. This raises the question: How much recall is lost due to the constraints?

To address this question, we analyzed Wu and Weld's set of 300 sentences from a set of random Web pages, manually identifying all verb-based relationships between noun phrase pairs. This resulted in a set of 327 relation phrases. For each relation phrase, we checked whether it satisfies our constraints. We found that 85% of the relation phrases do satisfy the constraints. Of the remaining 15%, we identified some of the common cases where the constraints were violated, summarized in Table 3.

Many of the example relation phrases shown in Table 3 involve long-range dependencies between words in the sentence. These types of dependencies are not easily representable using a pattern over POS tags. A deeper syntactic analysis of the input sentence would provide a much more general language for modeling relation phrases. For example, one could create a model of relations expressed in

	Binary Verbal Relation Phrases
85%	Satisfy Constraints
8%	Non-Contiguous Phrase Structure Coordination: X is produced and maintained by Y Multiple Args: X was founded in 1995 by Y Phrasal Verbs: X turned Y off
4%	Relation Phrase Not Between Arguments Intro. Phrases: Discovered by Y, X ... Relative Clauses: ... the Y that X discovered
3%	Do Not Match POS Pattern Interrupting Modifiers: X has a lot of faith in Y Infinitives: X to attack Y

Table 3: Approximately 85% of the binary verbal relation phrases in a sample of Web sentences satisfy our constraints.

terms of dependency parse features that would capture the non-contiguous relation phrases in Table 3. Previous work has shown that dependency paths do indeed boost the recall of relation extraction systems (Wu and Weld, 2010; Mintz et al., 2009). While using dependency path features allows for a more flexible model of relations, it significantly increases processing time, which is problematic for Web-scale extraction. Further, we have found that this increased recall comes at the cost of lower precision on Web text (see Section 5).

The results in Table 3 are similar to Banko and Etzioni's findings that a set of eight POS patterns cover a large fraction of binary verbal relation phrases. However, their analysis was based on a set of sentences known to contain either a company acquisition or birthplace relationship, while our results are on a random sample of Web sentences. We applied Banko and Etzioni's verbal patterns to our random sample of 300 Web sentences, and found that they cover approximately 69% of the relation phrases in the corpus. The gap in recall between this and the 85% shown in Table 3 is largely due to LVC relation phrases (*made a deal with*) and phrases containing multiple verbs (*refuses to return to*), which their patterns do not cover.

In sum, our model is by no means complete. However, we have empirically shown that the majority of binary verbal relation phrases in a sample of Web sentences are captured by our model. By focusing on this subset of language, our model can be used to perform Open IE at significantly higher precision than before.

4 REVERB

This section introduces REVERB, a novel open extractor based on the constraints defined in the previous section. REVERB first identifies relation phrases that satisfy the syntactic and lexical constraints, and then finds a pair of NP arguments for each identified relation phrase. The resulting extractions are then assigned a confidence score using a logistic regression classifier.

This algorithm differs in three important ways from previous methods (Section 2). First, the relation phrase is identified "holistically" rather than word-by-word. Second, potential phrases are filtered based on statistics over a large corpus (the implementation of our lexical constraint). Finally, REVERB is "relation first" rather than "arguments first", which enables it to avoid a common error made by previous methods—confusing a noun in the relation phrase for an argument, *e.g.* the noun *deal* in *made a deal with*.

4.1 Extraction Algorithm

REVERB takes as input a POS-tagged and NP-chunked sentence and returns a set of (x, r, y) extraction triples.[2] Given an input sentence s, REVERB uses the following extraction algorithm:

1. **Relation Extraction:** For each verb v in s, find the longest sequence of words r_v such that (1) r_v starts at v, (2) r_v satisfies the syntactic constraint, and (3) r_v satisfies the lexical constraint. If any pair of matches are adjacent or overlap in s, merge them into a single match.
2. **Argument Extraction:** For each relation phrase r identified in Step 1, find the nearest noun phrase x to the left of r in s such that x is not a relative pronoun, WHO-adverb, or existential "there". Find the nearest noun phrase y to the right of r in s. If such an (x, y) pair could be found, return (x, r, y) as an extraction.

We check whether a candidate relation phrase r_v satisfies the syntactic constraint by matching it against the regular expression in Figure 1.

[2] REVERB uses OpenNLP for POS tagging and NP chunking: `http://opennlp.sourceforge.net/`

To determine whether r_v satisfies the lexical constraint, we use a large dictionary D of relation phrases that are known to take many distinct arguments. In an offline step, we construct D by finding all matches of the POS pattern in a corpus of 500 million Web sentences. For each matching relation phrase, we heuristically identify its arguments (as in Step 2 above). We set D to be the set of all relation phrases that take at least k distinct argument pairs in the set of extractions. In order to allow for minor variations in relation phrases, we normalize each relation phrase by removing inflection, auxiliary verbs, adjectives, and adverbs. Based on experiments on a held-out set of sentences, we found that a value of $k = 20$ works well for filtering out overspecified relations. This results in a set of approximately 1.7 million distinct normalized relation phrases, which are stored in memory at extraction time.

As an example of the extraction algorithm in action, consider the following input sentence:

> Hudson was born in Hampstead, which is a suburb of London.

Step 1 of the algorithm identifies three relation phrases that satisfy the syntactic and lexical constraints: *was*, *born in*, and *is a suburb of*. The first two phrases are adjacent in the sentence, so they are merged into the single relation phrase *was born in*. Step 2 then finds an argument pair for each relation phrase. For *was born in*, the nearest NPs are *(Hudson, Hampstead)*. For *is a suburb of*, the extractor skips over the NP *which* and chooses the argument pair *(Hampstead, London)*. The final output is

e_1: *(Hudson, was born in, Hampstead)*
e_2: *(Hampstead, is a suburb of, London)*.

4.2 Confidence Function

The extraction algorithm in the previous section has high recall, but low precision. Like with previous open extractors, we want way to trade recall for precision by tuning a confidence threshold. We use a logistic regression classifier to assign a confidence score to each extraction, which uses the features shown in Table 4. All of these features are efficiently computable and relation independent. We trained the confidence function by manually labeling the extractions from a set of $1,000$ sentences from the Web and Wikipedia as correct or incorrect.

Weight	Feature
1.16	(x, r, y) covers all words in s
0.50	The last preposition in r is *for*
0.49	The last preposition in r is *on*
0.46	The last preposition in r is *of*
0.43	$len(s) \leq 10$ words
0.43	There is a WH-word to the left of r
0.42	r matches VW*P from Figure 1
0.39	The last preposition in r is *to*
0.25	The last preposition in r is *in*
0.23	10 words $< len(s) \leq 20$ words
0.21	s begins with x
0.16	y is a proper noun
0.01	x is a proper noun
-0.30	There is an NP to the left of x in s
-0.43	20 words $< len(s)$
-0.61	r matches V from Figure 1
-0.65	There is a preposition to the left of x in s
-0.81	There is an NP to the right of y in s
-0.93	Coord. conjunction to the left of r in s

Table 4: REVERB uses these features to assign a confidence score to an extraction (x, r, y) from a sentence s using a logistic regression classifier.

Previous open extractors require labeled training data to learn a model of relations, which is then used to extract relation phrases from text. In contrast, REVERB uses a specified model of relations for extraction, and requires labeled data only for assigning confidence scores to its extractions. Learning a confidence function is a much simpler task than learning a full model of relations, using two orders of magnitude fewer training examples than TEXTRUNNER or WOE.

4.3 TEXTRUNNER-R

The model of relation phrases used by REVERB is specified, but could a TEXTRUNNER-like system learn this model from training data? While it is difficult to answer such a question for all possible permutations of features sets, training examples, and learning biases, we demonstrate that TEXTRUNNER itself cannot learn REVERB's model even when re-trained using the output of REVERB as labeled training data. The resulting system, TEXTRUNNER-R, uses the same feature representation as TEXTRUNNER, but different parameters, and a different set of training examples.

To generate positive instances, we ran REVERB

on the Penn Treebank, which is the same dataset that TEXTRUNNER is trained on. To generate negative instances from a sentence, we took each noun phrase pair in the sentence that does not appear as arguments in a REVERB extraction. This process resulted in a set of $67,562$ positive instances, and $356,834$ negative instances. We then passed these labeled examples to TEXTRUNNER's training procedure, which learns a linear-chain CRF using closed-class features like POS tags, capitalization, punctuation, *etc.* TEXTRUNNER-R uses the argument-first extraction algorithm described in Section 2.

5 Experiments

We compare REVERB to the following systems:

- REVERB$^{\neg lex}$ - The REVERB system described in the previous section, but without the lexical constraint. REVERB$^{\neg lex}$ uses the same confidence function as REVERB.
- TEXTRUNNER - Banko and Etzioni's 2008 extractor, which uses a second order linear-chain CRF trained on extractions heuristically generated from the Penn Treebank. TEXTRUNNER uses shallow linguistic features in its CRF, which come from the same POS tagger and NP-chunker that REVERB uses.
- TEXTRUNNER-R - Our modification to TEXTRUNNER, which uses the same extraction code, but with a model of relations trained on REVERB extractions.
- WOE^{pos} - Wu and Weld's modification to TEXTRUNNER, which uses a model of relations learned from extractions heuristically generated from Wikipedia.
- WOE^{parse} - Wu and Weld's parser-based extractor, which uses a large dictionary of dependency path patterns learned from heuristic extractions generated from Wikipedia.

Each system is given a set of sentences as input, and returns a set of binary extractions as output. We created a test set of 500 sentences sampled from the Web, using Yahoo's random link service.[3] After running each extractor over the input sentences, two human judges independently evaluated each extraction as correct or incorrect. The judges reached agreement on 86% of the extractions, with an agreement score of $\kappa = 0.68$. We report results on the subset of the data where the two judges concur.

[3] http://random.yahoo.com/bin/ryl

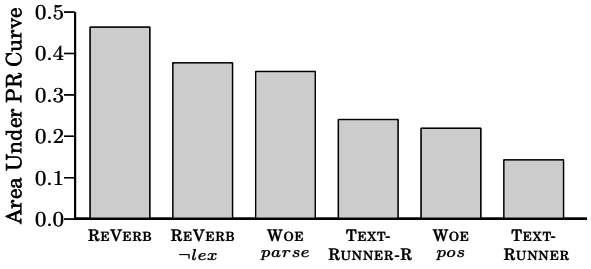


Figure 2: REVERB outperforms state-of-the-art open extractors, with an AUC more than twice that of TEXTRUNNER or WOE^{pos}, and 38% higher than WOE^{parse}.

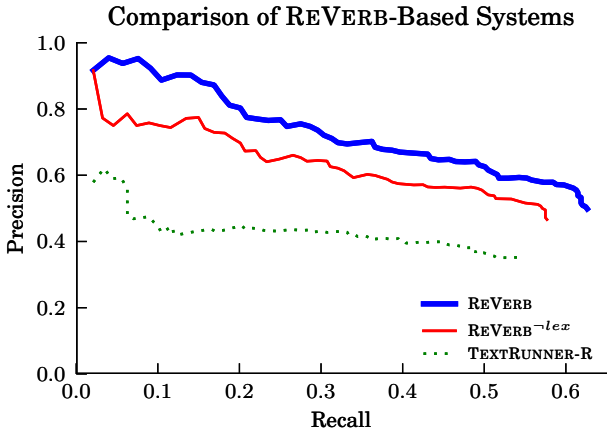


Figure 3: The lexical constraint gives REVERB a boost in precision and recall over REVERB$^{\neg lex}$. TEXTRUNNER-R is unable to learn the model used by REVERB, which results in lower precision and recall.

The judges labeled uninformative extractions conservatively. That is, if critical information was dropped from the relation phrase but included in the second argument, it is labeled correct. For example, both the extractions *(Ackerman, is a professor of, biology)* and *(Ackerman, is, a professor of biology)* are considered correct.

Each system returns confidence scores for its extractions. For a given threshold, we can measure the precision and recall of the output. Precision is the fraction of returned extractions that are correct. Recall is the fraction of correct extractions in

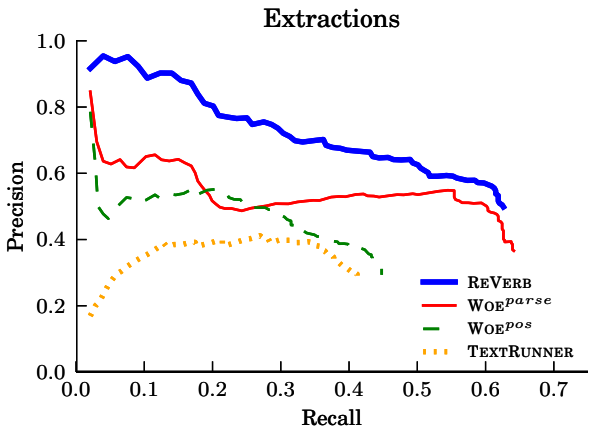


Figure 4: REVERB achieves significantly higher precision than state-of-the-art Open IE systems, and comparable recall to WOE^{parse}.

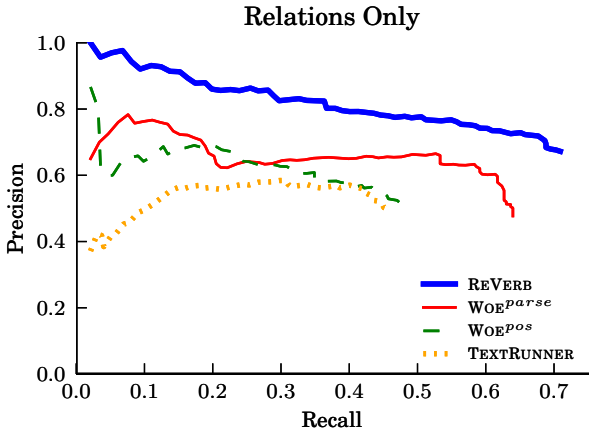


Figure 5: On the subtask of identifying relations phrases, REVERB is able to achieve even higher precision and recall than other systems.

the corpus that are returned. We use the total number of extractions labeled as correct by the judges as our measure of recall for the corpus. In order to avoid double-counting, we treat extractions that differ superficially (*e.g.*, different punctuation or dropping inessential modifiers) as a single extraction. We compute a precision-recall curve by varying the confidence threshold, and then compute the area under the curve (AUC).

5.1 Results

Figure 2 shows the AUC of each system. REVERB achieves an AUC that is 30% higher than WOE^{parse} and is more than double the AUC of WOE^{pos} or TEXTRUNNER. The lexical constraint provides a significant boost in performance, with REVERB achieving an AUC 23% higher than REVERB$^{\neg lex}$. REVERB proves to be a useful source of training data, with TEXTRUNNER-R having an AUC 71% higher than TEXTRUNNER and performing on par with WOE^{pos}. From the training data, TEXTRUNNER-R was able to learn a model that predicts contiguous relation phrases, but still returned incoherent relation phrases (*e.g.*, starting with a preposition) and overspecified relation phrases. These errors are due to TEXTRUNNER-R overfitting the training data and not having access to the lexical constraint.

Figure 3 shows the precision-recall curves of the systems introduced in this paper. TEXTRUNNER-R has much lower precision than REVERB and REVERB$^{\neg lex}$ at all levels of recall. The lexical constraint gives REVERB a boost in precision over REVERB$^{\neg lex}$, reducing overspecified extractions from 20% of REVERB$^{\neg lex}$'s output to 1% of REVERB's. The lexical constraint also boosts recall over REVERB$^{\neg lex}$, since REVERB is able to find a correct relation phrase where REVERB$^{\neg lex}$ finds an overspecified one.

Figure 4 shows the precision-recall curves of REVERB and the external systems. REVERB has much higher precision than the other systems at nearly all levels of recall. In particular, more than 30% of REVERB's extractions are at precision 0.8 or higher, compared to virtually none for the other systems. WOE^{parse} achieves a slightly higher recall than REVERB (0.62 versus 0.64), but at the cost of lower precision.

In order to highlight the role of the relational model of each system, we also evaluate their performance on the subtask of extracting just the relation phrases from the input text. Figure 5 shows the precision-recall curves for each system on the relation phrase-only evaluation. In this case, REVERB has both higher precision and recall than the other systems.

REVERB's biggest improvement came from the elimination of incoherent extractions. Incoherent extractions were a large fraction of the errors made by previous systems, accounting for approximately 13% of TEXTRUNNER's extractions, 15% of WOE^{pos}'s, and 30% of WOE^{parse}'s. Uninformative

REVERB - Incorrect Extractions	
65%	Correct relation phrase, incorrect arguments
16%	N-ary relation
8%	Non-contiguous relation phrase
2%	Imperative verb
2%	Overspecified relation phrase
7%	Other, including POS/chunking errors

Table 5: The majority of the incorrect extractions returned by REVERB are due to errors in argument extraction.

REVERB - Missed Extractions	
52%	Could not identify correct arguments
23%	Relation filtered out by lexical constraint
17%	Identified a more specific relation
8%	POS/chunking error

Table 6: The majority of extractions that were missed by REVERB were cases where the correct relation phrase was found, but the arguments were not correctly identified.

extractions had a smaller effect on other systems' precision, accounting for 4% of WOE^{parse}'s extractions, 5% of WOE^{pos}'s, and 7% of TEXTRUNNER's, while only appearing in 1% of REVERB's extractions. REVERB's reduction in uninformative extractions resulted in a boost in recall, capturing many LVC relation phrases missed by other systems (like those shown in Table 2).

To test the systems' speed, we ran each extractor on a set of 100,000 sentences using a Pentium 4 machine with 4GB of RAM. The processing times were 16 minutes for REVERB, 21 minutes for TEXTRUNNER, 21 minutes for WOE^{pos}, and 11 hours for WOE^{parse}. The times for REVERB, TEXTRUNNER, and WOE^{pos} are all approximately the same, since they all use the same POS-tagging and NP-chunking software. WOE^{parse} processes each sentence with a dependency parser, resulting in much longer processing time.

5.2 REVERB Error Analysis

To better understand the limitations of REVERB, we performed a detailed analysis of its errors in precision (incorrect extractions returned by REVERB) and its errors in recall (correct extractions that REVERB missed).

Table 5 summarizes the types of incorrect extractions that REVERB returns. We found that 65% of the incorrect extractions returned by REVERB were cases where a relation phrase was correctly identified, but the argument-finding heuristics failed. The remaining errors were cases where REVERB extracted an incorrect relation phrase. One common mistake that REVERB made was extracting a relation phrase that expresses an n-ary relationship via a ditransitive verb. For example, given the sentence "I gave him 15 photographs," REVERB extracts *(I, gave, him)*. These errors are due to the fact that REVERB only models binary relations.

Table 6 summarizes the correct extractions that were extracted by other systems and were not extracted by REVERB. As with the false positive extractions, the majority of false negatives (52%) were due to the argument-finding heuristics choosing the wrong arguments, or failing to extract all possible arguments (in the case of coordinating conjunctions). Other sources of failure were due to the lexical constraint either failing to filter out an overspecified relation phrase or filtering out a valid relation phrase. These errors hurt both precision and recall, since each case results in the extractor overlooking a correct relation phrase and choosing another.

5.3 Evaluation At Scale

Section 5.1 shows that REVERB outperforms existing Open IE systems when evaluated on a sample of sentences. Previous work has shown that the frequency of an extraction in a large corpus is useful for assessing the correctness of extractions (Downey et al., 2005). Thus, it is possible *a priori* that REVERB's gains over previous systems will diminish when extraction frequency is taken into account.

In fact, we found that REVERB's advantage over TEXTRUNNER when run at scale is qualitatively similar to its advantage on single sentences. We ran both REVERB and TEXTRUNNER on Banko and Etzioni's corpus of 500 million Web sentences and examined the effect of redundancy on precision.

As Downey's work predicts, precision increased in both systems for extractions found multiple times, compared with extractions found only once. However, REVERB had higher precision than

TEXTRUNNER at all frequency thresholds. In fact, REVERB's frequency 1 extractions had a precision of 0.75, which TEXTRUNNER could not approach even with frequency 10 extractions, which had a precision of 0.34. Thus, REVERB is able to return more correct extractions at a higher precision than TEXTRUNNER, even when redundancy is taken into account.

6 Conclusions and Future Work

The paper's contributions are as follows:

- We have identified and analyzed the problems of incoherent and uninformative extractions for Open IE systems, and shown their prevalence for systems such as TEXTRUNNER and WOE.
- We articulated general, easy-to-enforce constraints on binary, verb-based relation phrases in English that ameliorate these problems and yield richer and more informative relations (see, for example, Table 2).
- Based on these constraints, we designed, implemented, and evaluated the REVERB extractor, which substantially outperforms previous Open IE systems in both recall and precision.
- We make REVERB and the data used in our experiments available to the research community.[4]

In future work, we plan to explore utilizing our constraints to improve the performance of learned CRF models. Roth *et al.* have shown how to incorporate constraints into CRF learners (Roth and Yih, 2005). It is natural, then, to consider whether the combination of heuristically labeled training examples, CRF learning, and our constraints will result in superior performance. The error analysis in Section 5.2 also suggests natural directions for future work. For instance, since many of REVERB's errors are due to incorrect arguments, improved methods for argument extraction are in order.

Acknowledgments

We would like to thank Mausam, Dan Weld, Yoav Artzi, Luke Zettlemoyer, members of the KnowItAll group, and the anonymous reviewers for their helpful comments. This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-08-1-0431, and DARPA contract FA8750-09-C-0179, and carried out at the University of Washington's Turing Center.

[4] http://reverb.cs.washington.edu

References

David J. Allerton. 2002. *Stretched Verb Constructions in English*. Routledge Studies in Germanic Linguistics. Routledge (Taylor and Francis), New York.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36, Columbus, Ohio, June. Association for Computational Linguistics.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *In the Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676, January.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of ACL*, Portland, OR.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2010. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 52–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Doug Downey, Oren Etzioni, and Stephen Soderland. 2005. A probabilistic model of redundancy in information extraction. In *IJCAI*, pages 1034–1041.

Gregory Grefenstette and Simone Teufel. 1995. Corpus-based method for automatic identification of support verbs for nominalizations. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 98–103, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 286–295, Stroudsburg, PA, USA. Association for Computational Linguistics.

Holmer Hemsen Kathrin Eichler and Gnter Neumann. 2008. Unsupervised relation extraction from web documents. In *LREC*. http://www.lrec-conf.org/proceedings/lrec2008/.

J. Kim and D. Moldovan. 1993. Acquisition of semantic patterns for information extraction from corpora. In *Procs. of Ninth IEEE Conference on Artificial Intelligence for Applications*, pages 171–176.

Dekang Lin and Patrick Pantel. 2001. DIRT-Discovery of Inference Rules from Text. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining(KDD-01)*, pages pp. 323–328.

Thomas Lin, Mausam, and Oren Etzioni. 2010. Identifying Functional Relations in Web Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1266–1276, Cambridge, MA, October. Association for Computational Linguistics.

Paul Kingsbury Martha and Martha Palmer. 2002. From treebank to propbank. In *In Proceedings of LREC-2002*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 1003–1011, Morristown, NJ, USA. Association for Computational Linguistics.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

E. Riloff. 1996. Automatically constructing extraction patterns from untagged text. In *Procs. of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A Latent Dirichlet Allocation Method for Selectional Preferences. In *ACL*.

Dan Roth and Wen-tau Yih. 2005. Integer linear programming inference for conditional random fields. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 736–743, New York, NY, USA. ACM.

Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1088–1098, Stroudsburg, PA, USA. Association for Computational Linguistics.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 731–738, Morristown, NJ, USA. Association for Computational Linguistics.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive Information Extraction using Unrestricted Relation Discovery. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 304–311, New York City, USA, June. Association for Computational Linguistics.

Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni. 2010. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102.

S. Soderland. 1999. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272.

Lucia Specia and Enrico Motta. 2006. M.: A hybrid approach for extracting semantic relations from texts. In *In. Proceedings of the 2 nd Workshop on Ontology Learning and Population*, pages 57–64.

Suzanne Stevenson, Afsaneh Fazly, and Ryan North. 2004. Statistical measures of the semi-productivity of light verb constructions. In *2nd ACL Workshop on Multiword Expressions*, pages 1–8.

M. Stevenson. 2004. An unsupervised WordNet-based algorithm for relation extraction. In *Proceedings of the "Beyond Named Entity" workshop at the Fourth International Conference on Language Resources and Evalutaion (LREC'04)*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Morristown, NJ, USA. Association for Computational Linguistics.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. StatSnowball: a statistical approach to extracting entity relationships. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 101–110, New York, NY, USA. ACM.

Active Learning with Amazon Mechanical Turk

Florian Laws Christian Scheible Hinrich Schütze
Institute for Natural Language Processing
Universität Stuttgart
{lawsfn, scheibcn}@ims.uni-stuttgart.de

Abstract

Supervised classification needs large amounts of annotated training data that is expensive to create. Two approaches that reduce the cost of annotation are *active learning* and *crowdsourcing*. However, these two approaches have not been combined successfully to date. We evaluate the utility of active learning in crowdsourcing on two tasks, named entity recognition and sentiment detection, and show that active learning outperforms random selection of annotation examples in a noisy crowdsourcing scenario.

1 Introduction

Supervised classification is the predominant technique for a large number of natural language processing (NLP) tasks. The large amount of labeled training data that supervised classification relies on is time-consuming and expensive to create, especially when experts perform the data annotation. Recently, crowdsourcing services like Amazon Mechanical Turk (MTurk) have become available as an alternative that offers acquisition of non-expert annotations at low cost. MTurk is a software service that outsources small annotation tasks – called *HITs* – to a large group of freelance workers. The cost of MTurk annotation is low, but a consequence of using non-expert annotators is much lower annotation quality. This requires strategies for quality control of the annotations.

Another promising approach to the data acquisition bottleneck for supervised learning is active learning (AL). AL reduces annotation effort by setting up an annotation loop where, starting from a small seed set, only the maximally informative examples are chosen for annotation. With these annotated examples, the classifier is then retrained to again select more informative examples for further annotation. In general, AL needs a lot fewer annotations to achieve a desired performance level than random sampling.

AL has been successfully applied to a number of NLP tasks such as part-of-speech tagging (Ringger et al., 2007), parsing (Osborne and Baldridge, 2004), text classification (Tong and Koller, 2002), sentiment detection (Brew et al., 2010), and named entity recognition (NER) (Tomanek et al., 2007). Until recently, most AL studies focused on simulating the annotation process by using already available gold standard data. In reality, however, human annotators make mistakes, leading to noise in the annotations. For this reason, some authors have questioned the applicability of AL to noisy annotation scenarios such as MTurk (Baldridge and Palmer, 2009; Rehbein et al., 2010).

AL and crowdsourcing are complementary approaches: AL reduces the number of annotations used while crowdsourcing reduces the cost per annotation. Combined, the two approaches could substantially lower the cost of creating training sets.

Our main contribution in this paper is that we show for the first time that AL is significantly better than randomly selected annotation examples in a real crowdsourcing annotation scenario. Our experiments directly address two tasks, named entity recognition and sentiment detection, but our

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1546–1556,
Edinburgh, Scotland, UK, July 27–31, 2011.

evidence suggests that AL is of general benefit in crowdsourcing. We also show that the effectiveness of MTurk annotation with AL can be further enhanced by using two techniques that increase label quality: *adaptive voting* and *fragment recovery*.

2 Related Work

2.1 Crowdsourcing

Pioneered by Snow et al. (2008), Crowdsourcing, especially using MTurk, has become a widely used service in the NLP community. A number of studies have looked at crowdsourcing for NER. Voyer et al. (2010) use a combination of expert and crowdsourced annotations. Finin et al. (2010) annotate Twitter messages – short sequences of words – and this is reflected in their vertically oriented user interface. Lawson et al. (2010) choose an annotation interface where annotators have to drag the mouse to select entities. Carpenter and Poesio (2010) argue that dragging is less convenient for workers than marking tokens.

These papers do not address AL in crowdsourcing. Another important difference is that previous studies on NER have used data sets for which no "linguistic" gold annotation is available. In contrast, we reannotate the CoNLL-2003 English NER dataset. This allows us to conduct a detailed comparison of MTurk AL to conventional expert annotation.

2.2 Active Learning with Noisy Labels

Hachey et al. (2005) were among the first to investigate the effect of actively sampled instances on agreement of labels and annotation time. They demonstrate applicability of AL when annotators are trained experts. This is an important result. However, AL depends on accurate assessments of uncertainty and informativeness and such an accurate assessment is made more difficult if labels are noisy as is the case in crowdsourcing. For this reason, the problem of AL performance with noisy labels has become a topic of interest in the AL community. Rehbein et al. (2010) investigate AL with human expert annotators for word sense disambiguation, but do not find convincing evidence that AL reduces annotation cost in a realistic (non-simulated) annotation scenario. Brew et al. (2010) carried out experiments on sentiment active learning through crowdsourcing. However, they use a small set of volunteer labelers instead of anonymous paid workers.

Donmez and Carbonell (2008) propose a method to choose annotators from a set of noisy annotators. However, in a crowdsourcing scenario, it is not possible to ask specific annotators for a label, as crowdsourcing workers join and leave the site. Furthermore, they only evaluate their approach in simulations. We use the actual labels of human annotators to avoid the risk of unrealistic assumptions when modeling annotators.

We are not aware of any study that shows that AL is significantly better than a simple baseline of having annotators annotate randomly selected examples in a highly noisy annotation setting like crowdsourcing. While AL generally is superior to this baseline in simulated experiments, it is not clear that this result carries over to crowdsourcing annotation. Crowdsourcing differs in a number of ways from simulated experiments: the difficulty and annotation consistency of examples drawn by AL differs from that drawn by random sampling; crowdsourcing labels are noisy; and because of the noisiness of labels statistical classifiers behave differently in simulated and real annotation experiments.

3 Annotation System

One fundamental design criterion for our annotation system was the ability to select examples *in real time* to support, e.g., the interactive annotation experiments presented in this paper. Thus, we could not use the standard MTurk workflow or services like CrowdFlower.[1]

We therefore designed our own system for annotation experiments. It consists of a two-tiered application architecture. The frontend tier is a web application that serves two purposes. First, the administrator can manage annotation experiments using a web interface and publish annotation tasks associated with an experiment on MTurk. The frontend also provides tools for efficient review of the received answers. Second, the frontend web application presents annotation tasks to MTurk workers. Because we wanted to implement interactive annotation experiments, we used the "external question"

[1] `http://crowdflower.com/`

feature of MTurk. An external question contains an URL to our frontend web application, which is queried when a worker views an annotation task. Our frontend then in turn queries our backend component for an example to be annotated and renders it in HTML.

The backend component is responsible for selection of an example to be annotated in response to a worker's request for an annotation task. The backend implements a diverse choice of random and active selection strategies as well as the multilabeling strategies described in section 3.2. The backend component runs as a standalone server and is queried by the frontend via REST-like HTTP calls.

For the NER task, we present one sentence per HIT, segmented into tokens, with a select box underneath each token containing the classes. The definition of the classes is based on the CoNLL-2003 annotation guidelines (Tjong Kim Sang and De Meulder, 2003). Examples were given for every class. Annotators are forced to make a selection for uppercase tokens. Lowercase tokens are prelabeled with "O" (no named entity), but annotators are encouraged to change this label if the token is in fact part of an entity phrase.

For sentiment annotation, we found in preliminary experiments that using simple radio button selection for the choice of the document label (`positive` or `negative`) leads to a very high amount of spam submissions, taking the overall classification accuracy down to around 55%. We then designed a template that forced annotators to type the label as well as a randomly chosen word from the text. Individual label accuracy was around 75% in this scheme.

3.1 Concurrent example selection

AL works by setting up an interactive annotation loop where at each iteration, the most informative example is selected for annotation. We use a pool-based AL setup where the most informative example is selected from a pool of unlabeled examples. Informativeness is calculated as uncertainty (Lewis and Gale, 1994) using the margin metric (Schein and Ungar, 2007). This metric chooses examples for which the margin of probabilities from the classifier between the two most probable classes is the smallest:

$$M_n = |\hat{P}(c_1|x_n) - \hat{P}(c_2|x_n)|$$

Here, x_n is the instance to be classified, c_1 and c_2 are the two most likely classes, and $\hat{P}$ the classifier's estimate of probability.

For NER, the margins of the tokens are averaged to get an uncertainty assessment of the sentence. For sentiment, whole documents are classified, thus uncertainties can be used directly.

After annotation, the selected example is removed from the unlabeled pool and, together with its label(s), added to the set of labeled examples. The classifier is then retrained on the labeled examples and the informativeness of the remaining examples in the pool is re-evaluated.

Depending on the classifier and the sizes of pool and labeled set, retraining and reevaluation can take some time. To minimize wait times, traditional AL implementations select examples in batches of the n most informative examples. However, batch selection might not give the optimum selection (examples in a batch are likely to be redundant, see Brinker (2003)) and wait times can still occur between one batch and the next.

When performing annotation with MTurk, wait times are unacceptable. Thus, we perform the retraining and uncertainty rescoring concurrently with the annotation user interface. The unlabeled pool is stored in a priority queue that is ordered according to the examples' informativeness. The annotation user interface takes the most informative example from the pool and presents it to the annotator. The labeled example is then inserted into a second queue that feeds and updates retraining and rescoring processes. The pool queue then is resorted according to the new informativeness. In this way, annotation and example selection can run in parallel. This is similar to Haertel et al. (2010).

3.2 Adaptive voting and fragment recovery

MTurk labels often have a high error rate. A common strategy for improving label quality is to acquire *multiple labels* by different workers for each example and then consolidate the annotations into a single label of higher quality. To trade off number of annotated examples against quality of annotations, we adopt *adaptive voting*. It uses majority

			NER						Sentiment					
Budget			5820				6931		1130				1756	
			#train	F_1	cost/sent	w.-accuracy	#train	F_1	#train	Acc	cost/doc	w.-accuracy	#train	Acc
RS	1	S	5820	59.6	1.00	51.6	–	–	1130	70.4	1	74.8	–	–
	2	3-v	1624	61.4†	3.58	70.1	–	–	–	–	–	–	–	–
	3	5/4-v	1488	63.0†	3.91	71.6	1774	63.5	450	71.2	2.51	89.6	735	79.2
	4	5-v+f	1996	63.6†	2.91	71.8	2385	64.9†	–	–	–	–	–	–
AL	5	S	5820	67.0	1.00	66.5	–	–	1130	74.8	1	76.0	–	–
	6	3-v	1808	70.0†	3.21	78.8	–	–	–	–	–	–	–	–
	7	5/4-v	1679	70.4†	3.46	79.6	1966	70.6	455	77.4	2.48	89.0	715	81.8
	8	5-v+f	2165	70.5	2.68	79.3	2691	71.2	–	–	–	–	–	–

Table 1: For NER, active learning consistently beats random sampling on MTurk. NER F_1 evaluated on CoNLL test set A. #train = number of sentences in training set, S = single, 3-v = 3-voting, 5/4-voting = 5- and 4-voting for NER and sentiment resp., +f = using fragments; sentiment budget 1130 for run 1, sentiment budget 1756 averaged over 2 runs.

voting and is adaptive in the number of repeated annotations. For NER, a sentence is first annotated by two workers. Then majority voting is performed for each token individually. If there is a majority for every token that is greater than an agreement threshold α, the sentence is accepted with each token labeled with the majority label. Otherwise additional annotations are requested. A sentence is discarded if the number of repeated annotations exceeds a discard threshold d (d-voting).[2] We use the same scheme for sentiment; note that there is just one decision per HIT in this case, not several as in NER.

For NER, we also use *fragment recovery*: we salvage tokens with agreeing labels from discarded sentences. We cut the token sequence of a discarded sentence into several fragments that have agreeing tokens and discard only those parts that disagree. We then include these recovered fragments in the training data just like complete sentences.

Software release. Our active learning framework used can be downloaded at `http://www.ims.uni-stuttgart.de/~lawsfn/active/`.

4 Experiments, Results and Analysis

4.1 Experiments

In our NER experiments, we have workers reannotate the English corpus of the CoNLL-2003 NER shared task. We chose this corpus to be able to compare crowdsourced annotations with gold standard annotations. A HIT is one sentence and is offered for a base payment of $0.01. We filtered out answers that contained unannotated tokens or were obvious spam (e.g., all tokens labeled as MISC). For testing NER performance, we used a system based on conditional random fields with standard named entity features including the token itself, orthographic features like the occurrence of capitalization or special characters and context information about the tokens to the left/right of the current token.

The sentiment detection task was modeled after a well-known document analysis setup for sentiment classification, introduced by Pang et al. (2002). We use their corpus of 1000 positive and 1000 negative movie reviews and the Stanford maximum entropy classifier (Manning and Klein, 2003) to predict the sentiment label of each document d from a unigram representation of d. We randomly split this corpus into a test set of 500 reviews and an active learning pool of 1500 reviews. Each HIT consists of one document, valued at $0.01.

We compare random sampling (RS) and AL in combination with the proposed voting and fragment strategies with different parameters. We want to avoid rerunning experiments on MTurk over and over again, but on the other hand, we believe that using synthetic data for simulations is problematic because it is difficult to generate synthetic data with a realistic model of annotator errors. Thus, we logged a play-by-play record of the annotator interactions and labels. With this recording, we can then rerun strategies with different parameters.

We chose voting with at most $d = 5$ repetitions as

[2] It can take a while in this scheme for annotators to agree on a final annotation for a sentence. We make *tentative* labels of a sentence available to the classifier immediately and replace them with the final labels once voting is completed.

our main reannotation strategy for both random and active sampling for NER annotation. We use simple majority voting ($\alpha = .5$) for NER.

For sentiment, we set $d = 4$ and minimum agreement $\alpha = .75$ because the number of labels is smaller (2 vs. 5) and so random agreement is more likely for sentiment.

To get results for 3-voting NER, we take the recording and discard 5-voting votes not needed in 3-voting. This will result in roughly the same number of annotated sentences, but at a lower cost. This simulation of 3-voting is not exactly what would have happened on MTurk (e.g., the final vote on a sentence might be different, which then influences AL example selection), but we will assume that differences are rare and simulated and actual results are similar. The same considerations apply to single votes and to the sentiment experiments.

We always compare two strategies for the same annotation budget. For example, the *number of training sentences* in Table 1 differ in the two relevant columns, but all strategies compared use exactly *the same annotation budget* (5820, 6931, 1130, and 1756, respectively).

For the single annotation strategy, each interaction record contained only about 40% usable annotations, the rest were repeats. A comparison with the single annotation strategy over approx. 2000 sentences or 450 documents would not have been meaningful; therefore we chose to run an extra experiment with the single annotation strategy to match this up with the budgets of the voting strategies. The results are presented in two separate columns of Table 1 (budgets 6931 and 1756).

4.2 Results

For sentiment detection, *worker accuracy* or *label quality* – the percentage of correctly annotated documents – is 74.8. In contrast, for NER, worker accuracy – the percentage of non-O tokens annotated correctly – is only 51.6 (Table 1, line 1). This demonstrates the challenge of using MTurk for NLP annotation tasks. When we use single annotations of each sentence, NER performance is 59.6 F_1 for random sampling (line 1). When training with gold labels on the same sentences, the performance is 80.0 (not shown). This means we lose more than 20% due to poor worker accuracy. Adaptive voting and fragment recovery manage to recover a small part of the lost performance (lines 2–4); each of the three F_1 scores is significantly better than the one above it as indicated by † (Approximate Randomization Test (Noreen, 1989; Chinchor et al., 1993) as implemented by Padó (2006)).

Using AL turns out to be quite successful for NER performance. For single annotations, NER performance is 67.0 (line 5), an improvement of 7.4% compared to random sampling. Adaptive voting and fragment recovery again increase worker accuracy (lines 6–8) although total improvement of 3.5% (lines 8 vs. 5) is smaller than 4% for random (lines 4 vs. 1). The learning curves of AL vs. random in Figure 1 (top left) confirm this good result for AL. These learning curves are for tokens – not for sentences – to show that the reason for AL's better performance is not that it selects slightly longer sentences than random. In addition, the relative advantage of AL vs random decreases over time, which is typical of pool-based AL experiments.

We carried out two runs of the same experiment for sentiment to validate our first positive result since the difference between the two conditions is not as large as in NER (Figure 1, top right). After about 300 documents, active learning consistently outperforms random sampling. The first AL run performs better because of higher label quality in the beginning. The overall advantage of AL over random is lower than for NER because the set of labels is smaller in sentiment, making the classification task easier. Second, there is a large amount of simple lexical clues for detecting sentiment (cf. Wilson et al. (2005)). It is likely that some of them can be learned well through random sampling at first; however, active learning can gain accuracy over time because it selects examples with more difficult clues.

In Figure 1 (bottom), we compare single annotation with adaptive voting. The graphs show F_1 as a function of cost. Adaptive voting trades quantity of sampled sentences for quality of labels and thus incurs higher net costs per sentence. This results in a smaller dataset for a given budget, but this dataset is still more useful for classifier training. For NER (Figure 1, bottom left), the single annotation strategy has a faster start; so for small budgets, covering a somewhat larger portion of the sample space is beneficial. For larger budgets, however, quality of

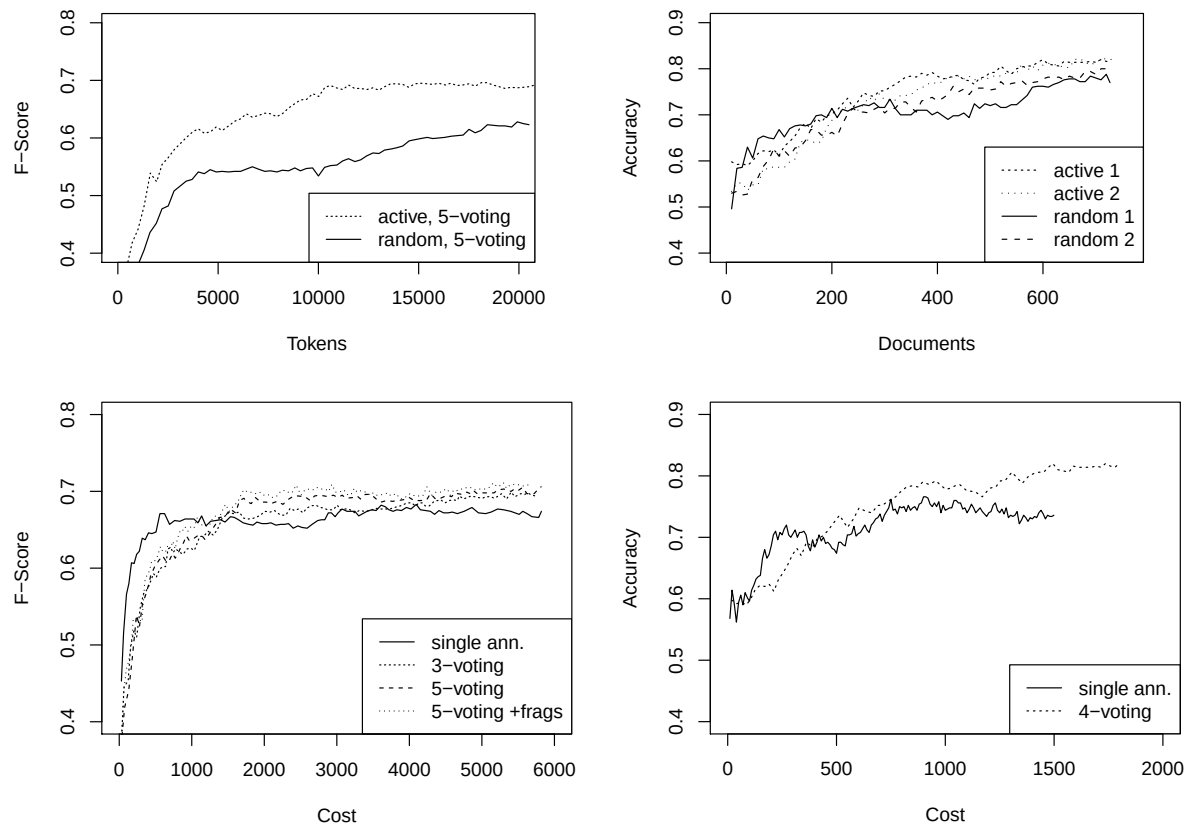


Figure 1: Top: Active learning vs. Random sampling for NER (left) and sentiment (right). Bottom: Active learning: adaptive voting vs. single annotation for NER (left) and sentiment (right).

the voted labels trumps quantity.

For sentiment (Figure 1, bottom right), results are similar: voting has no benefit initially, but as finding maximally informative examples to annotate becomes harder in later stages of learning, adaptive voting gains an advantage over single annotations.

The main result of the experiment is that active learning is better by about 7% F_1 than random sampling for NER and by 2.6% accuracy for sentiment (averaged over two runs at budget 1756). Adaptive voting further improves AL performance for both NER and sentiment.

4.3 Annotation time per token

Most AL work assumes constant cost per annotation unit. This assumption has been questioned because AL often selects hard examples that take longer to annotate (Hachey et al., 2005; Settles et al., 2008).

In annotation with MTurk, cost is not a function of annotation time because workers are paid a fixed amount per HIT. Nevertheless, annotation time plays a part in whether workers are willing to work on a given task for the offered reward. This is particularly problematic for NER since workers have to examine each token individually. We therefore investigate for NER whether the time MTurk workers spend on annotating sentences differs for random vs. AL.

We first compute median and mean annotation times and number of tokens per sentence:

	sec/sentence		tokens/sentence	
strategy	median	mean	all	required
random	17.2	33.1	15.0	3.4
AL	17.8	33.0	17.7	4.0

We see that most sentences are annotated in a very short time; but the mean is much larger than the median because there are outliers of up to eight minutes. AL tends to select slightly longer sentences as

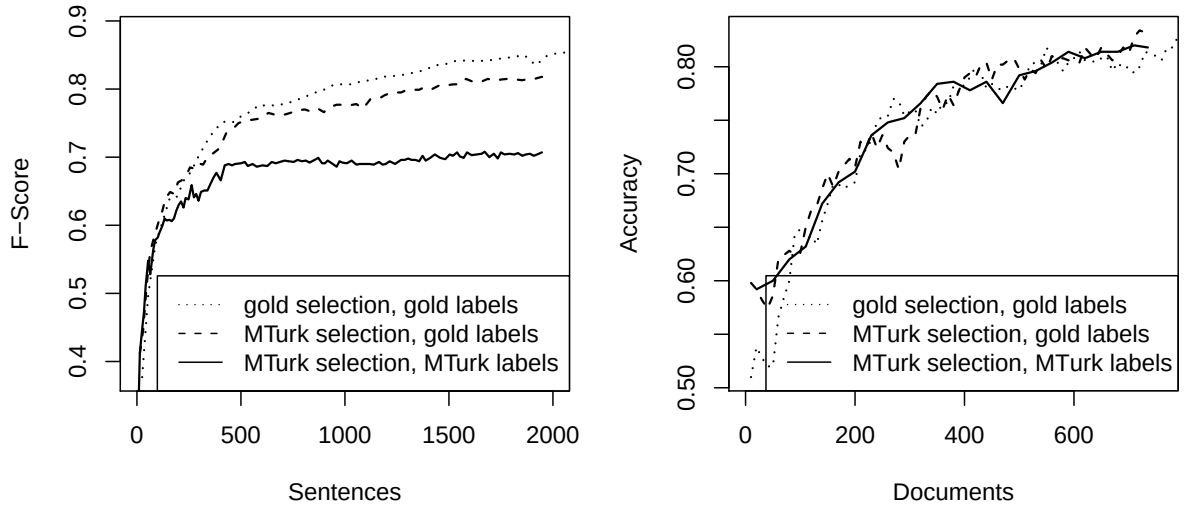


Figure 3: Performance on gold labels. Left: NER. Right: sentiment (run 1).

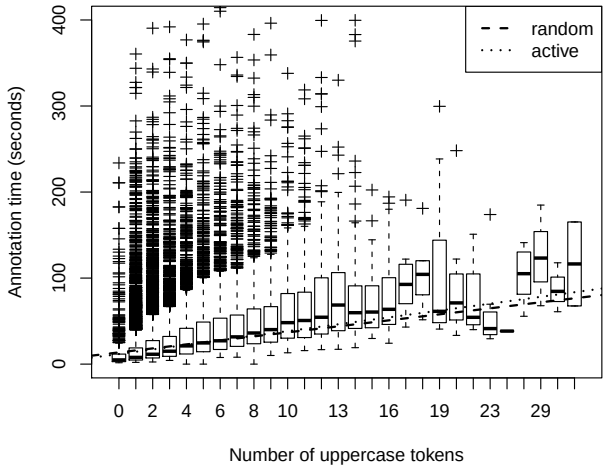


Figure 2: Annotation time vs. # uppercase tokens

well as sentences with slightly more uppercase tokens that require annotation.

In a more detailed analysis, we attempt to distinguish between (i) the effect of more uppercase ("annotation required") tokens vs. (ii) the effect of example difficulty. We fit a linear regression model to annotation time vs. the number of uppercase tokens. For the regression fit, we removed all annotation times > 60 seconds. Such long times indicate distraction of the worker and are not a reliable measure of difficulty.

Figure 2 shows the distribution of annotation times for both cases combined and the fitted models for each. The model estimated an annotation time of 2.3 secs for each required token for random vs. 2.7 secs for AL. We conclude that the difference in difficulty between sentences selected by random sampling vs. AL is small, but noticeable.

4.4 Influence of noise on the selection process

While NER performance for AL is much higher than for random sampling, it is still quite a bit lower than what is possible on gold labels. In the case of AL, there are two reasons why this happens: (i) The noisy labels negatively affect the classifier's ability to learn a good model that is used for classifying the test set. (ii) The noisy labels result in bad intermediate models that then select suboptimal examples to be annotated next. The AL selection process is "misled" by the noisy examples.

We conduct an experiment to determine the contribution of factors (i) and (ii) to the performance loss. First, we preserve the sequence of sentences chosen by our AL experiments on MTurk, with 5-voting for NER and 4-voting for sentiment but replace the noisy worker-provided labels by gold labels. The performance of classifiers trained on this sequence is the dashed line "MTurk selection, gold labels" in Figure 3 for NER (left) and sentiment (right).

Second, we compare with a traditional simulated AL experiment with gold labels. Here, the selection too is controlled by gold labels, so the selection has a noiseless classifier available for scoring and can perform optimal uncertainty selection. These are the

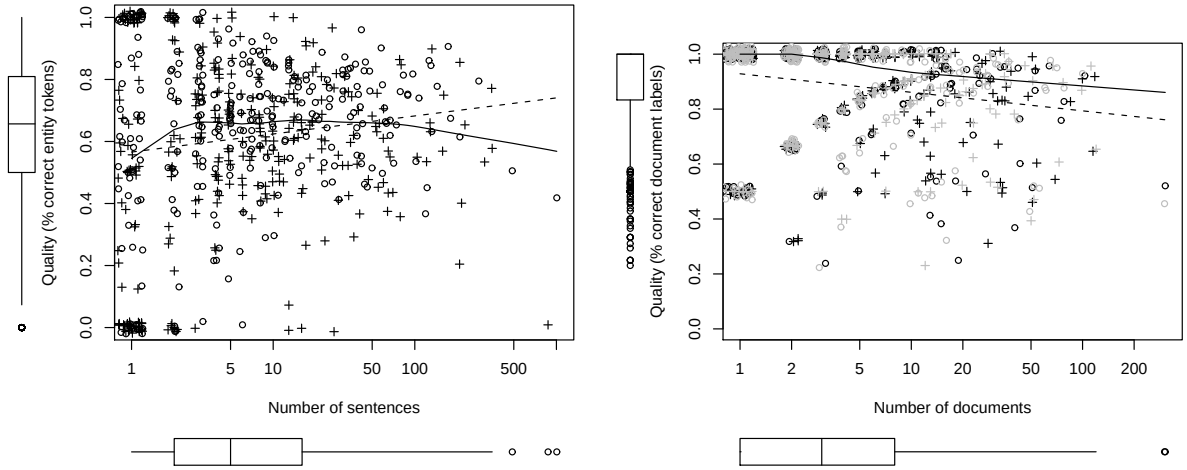


Figure 4: Worker accuracy vs. number of HITs. Each point corresponds to one worker ($\circ$ = active, $+$ =random sampling; black and grey for different runs). Left: NER. Right: Sentiment.

dotted lines "gold selection, gold labels" in Figure 3.

We used a batch-mode AL setup for this comparison experiment. For a fair comparison, we adjust the batchsize to be equal to the average *staleness* of a selected example in concurrent MTurk active learning. The staleness of an example is defined as the number of annotations the system has received, but not yet incorporated in the computation of an example's uncertainty score (Haertel et al., 2010).

For our concurrent NER system, the average staleness of an example was about 12 (min: 1, max: 40), for sentiment it was about 2. The figure for NER is higher than the number cited by Haertel et al. (2010) because there are more annotators accessing our system at the same time via MTurk but not as high for sentiment since documents are longer and retraining the sentiment classifier is faster. The average staleness of an example in a batch-mode system is half the batch size. Thus, we set the batch size of our comparison system to 25 for NER and to 4 for sentiment.

Returning to the two factors introduced above – (i) final effect of noise on test set performance vs. (ii) intermediate effect of noise on example selection – we see in Figure 3 that (i) has a large effect on NER whereas (ii) has a noticeable, but small effect.[3] For example, at 1966 sentences, F_1 scores are 70.6 (MTurk-MTurk), 81.4 (MTurk-gold) and 84.9 (gold-gold). This means that a performance difference of 10 points F_1 has to be attributed to noisy labels resulting in a worse final classifier (effect i), and another 3.5 points are lost due to sub-optimal example selection (effect ii).

For sentiment, the results are different. There is no clear difference between the three runs. We attribute this to the fact that the quality of the labels is higher in sentiment than in NER. Our initial experiments on sentiment were all negative (showing no improvement of AL compared to random) because label quality was too low. Only after we introduced the template described in Section 3 and used 4-voting with $\alpha = .75$ did we get positive results for AL. This leads to an overall label quality of about 90% (over all runs) which is so high that the difference to using gold labels is small if present at all.

5 Worker Quality

So far we have assumed that all workers provide annotations of the same quality. However, this is not the case. Figure 4 shows plots of worker accuracy as a function of worker productivity (number of annotated examples). Some workers submit only one or two HITs just to try out the task. For NER, the majority of workers submit between 5 and 10 sentences, with label qualities between 0.5 and 0.8. The chance level for correctness is around 0.25 (four

[3] Our comparison unit for NER is the sentence. We cannot compare on cost here since we do not know what the per-sentence cost of a "gold" expert annotation is.

different named entity categories for uppercase tokens). For sentiment, most workers submit 1 to 5 documents, with label qualities between 0.5 and 1. Chance level lies at around 0.5 (for two equally distributed labels).

While quality for highly productive workers is mediocre in our experiments, other researchers have found extremely bad quality for their most prolific workers (Callison-Burch, 2009). Some of these workers might be spammers who try to submit answers with automatic scripts. We encountered some spammers that our heuristics did not detect (shown in the bottom-right areas of Figure 4, left), but the voting mechanism was able to mitigate their negative influence.

Given the large variation in Figure 4, using worker quality in crowdsourcing for improved training set creation seems promising. We now test two such strategies for NER in an oracle setup.

5.1 Blocking low-quality workers

A simple approach is to refuse annotations from workers that have been determined to provide low quality answers. We simulated this strategy on NER data using oracle quality ratings. We chose NER because of its lower overall label quality. The results are presented in Figure 5 for random (a) and AL (b). For random, quality filtering with low cutoffs helps by removing bad annotations that likely come from spammers. While the voting strategy prevented a performance decrease with bad annotations, it needed to expend many extra annotations for correction. With filtering, these extra annotations become unnecessary and the system can learn faster. When low-quality workers are less active, as in the AL dataset, we find no meaningful performance increase for low cutoffs up to 0.4. For very high cutoffs (0.7), the beginning of the performance curve shows that further cost reductions can be achieved. However, we did not have enough recorded human annotations available to perform a simulation for the full budget.

5.2 Trusting high-quality workers

The complementary approach is to take annotations from highly rated workers at face value and immediately accept them as the correct label, *bypassing* the voting procedure. Bypassing saves the cost of repeated annotation of the same sentence. Figure 5 shows learning curves for two bypass thresholds on worker quality (measured as proportion of correct non-O tokens) for random (c) and AL (d). Bypassing performs surprisingly well. We find a steeper rise of the learning curve, meaning less cost for the same performance. Not only do we find substantial cost reductions, but also higher overall performance. We believe this is because high-quality annotations can sometimes be voted down by other annotations. If we can identify high-quality workers and directly use their annotations, this can be avoided.

These experiments are oracle experiments using gold data that is normally not available. In future work, we would like to repeat the experiments using methods for worker quality estimation (Ipeirotis et al., 2010; Donmez et al., 2009). For AL, the choice as to which labels are used (as a result of voting, bypassing or other) also has an influence on the selection. However, we had to keep the sequence of the selected sentences fixed in the simulations reported above. While our method of sample selection for AL proved to be quite robust even in the presence of noise, higher quality labels do have an influence on the sample selection (see section 4.4), so the improvement could be even better than indicated here.

5.3 Differences in quality between AL and random

The essence of AL is to select examples that are difficult to classify. As observed in our experiments on annotation time, this difficulty is reflected in the amount of time a human needs to work on examples selected through AL. Another effect to expect from difficulty could be lower annotation accuracy. We therefore examined the accuracies for each worker who contributed to both the AL and the random experiment. We found that in the NER task, the 20 workers in this group had a slightly higher (0.07) average quality for randomly selected examples. This difference is low and does not suggest a significant drop in accuracy for examples selected in AL.

6 Conclusion

We have investigated the use of AL in a real-life annotation experiment with human annotators instead of traditional simulations with gold labels for

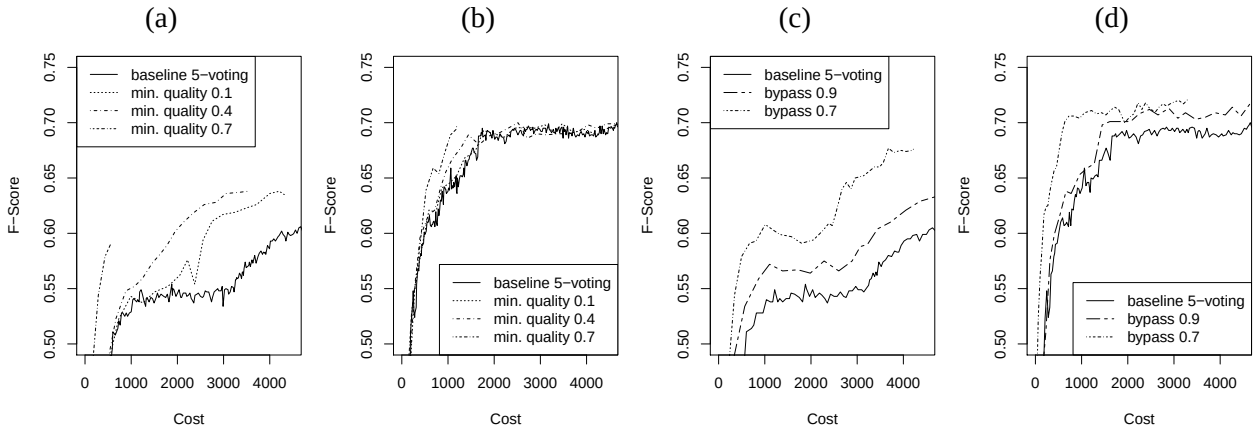


Figure 5: Blocking low-quality workers: (a) random, (b) AL. Bypass voting: (c) random, (d) AL.

named entity recognition and sentiment classification. The annotation was performed using MTurk in an AL framework that features concurrent example selection without wait times. We also evaluated two strategies, adaptive voting and fragment recovery, to improve label quality at low additional cost. We find that even for the relatively high noise levels of annotations gathered with MTurk, AL is successful, improving performance by +6.9 points F_1 compared to random sampling for NER and by +2.6% accuracy for sentiment. Furthermore, this performance level is reached at a smaller MTurk cost compared to random sampling. Thus AL not only reduces annotation costs, but also offers an improvement in absolute performance for these tasks. This is clear evidence that active learning and crowdsourcing are complementary methods for lowering annotation cost and should be used together in training set creation for natural language processing tasks.

We have also conducted oracle experiments that show that further performance gains and cost savings can be achieved by using information about worker quality. We plan to confirm these results by using estimates of quality in the future.

7 Acknowledgments

Florian Laws is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by his fellowship. Christian Scheible is supported by the Deutsche Forschungsgemeinschaft project Sonderforschungsbereich 732.

References

Jason Baldridge and Alexis Palmer. 2009. How well does active learning *actually* work? Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 296–305.

Anthony Brew, Derek Greene, and Pádraig Cunningham. 2010. Using crowdsourcing and active learning to track sentiment in online media. In *Proceeding of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 145–150.

Klaus Brinker. 2003. Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 59–66.

Chris Callison-Burch. 2009. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 286–295.

Bob Carpenter and Massimo Poesio. 2010. Models of data annotation. Tutorial at the seventh international conference on Language Resources and Evaluation (LREC 2010).

Nancy Chinchor, David D. Lewis, and Lynette Hirschman. 1993. Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). *Computational Linguistics*, 19(3):409–449.

Pinar Donmez and Jaime G. Carbonell. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 619–628.

Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. 2009. Efficiently learning the accuracy of la-

beling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 259–268.

Tim Finin, William Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88.

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *CoNLL '05: Proceedings of the 9th Conference on Computational Natural Language Learning*, pages 144–151.

Robbie Haertel, Paul Felt, Eric K. Ringger, and Kevin Seppi. 2010. Parallel active learning: Eliminating wait time with minimal staleness. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pages 33–41.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '10)*.

Nolan Lawson, Kevin Eustice, Mike Perkowitz, and Meliha Yetisgen-Yildiz. 2010. Annotating large email datasets for named entity recognition with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 71–79.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Christopher Manning and Dan Klein. 2003. Optimization, maxent models, and conditional estimation without magic. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Tutorials - Volume 5*, pages 8–8.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses: an introduction*. Wiley.

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 89–96.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

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

Ines Rehbein, Josef Ruppenhofer, and Alexis Palmer. 2010. Bringing active learning to life. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 949–957.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the Linguistic Annotation Workshop at ACL-2007*, pages 101–108.

Andrew Schein and Lyle Ungar. 2007. Active learning for logistic regression: An evaluation. *Machine Learning*, 68(3):235–265.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1069–1078.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL (CoNLL 2003)*, pages 142–147.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 486–495.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66.

Robert Voyer, Valerie Nygaard, Will Fitzgerald, and Hannah Copperman. 2010. A hybrid model for annotating named entity training corpora. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 243–246.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354.

Bootstrapped Named Entity Recognition for Product Attribute Extraction

Duangmanee (Pew) Putthividhya
eBay Inc.
2065 Hamilton Ave
San Jose, CA 95125
dputthividhya@ebay.com

Junling Hu
eBay Inc.
2065 Hamilton Ave
San Jose, CA 95125
juhu@ebay.com

Abstract

We present a named entity recognition (NER) system for extracting product attributes and values from listing titles. Information extraction from short listing titles present a unique challenge, with the lack of informative context and grammatical structure. In this work, we combine supervised NER with bootstrapping to expand the seed list, and output normalized results. Focusing on listings from eBay's clothing and shoes categories, our bootstrapped NER system is able to identify new brands corresponding to spelling variants and typographical errors of the known brands, as well as identifying novel brands. Among the top 300 new brands predicted, our system achieves 90.33% precision. To output normalized attribute values, we explore several string comparison algorithms and found n-gram substring matching to work well in practice.

1 Introduction

Traditional named entity recognition (NER) task has expanded beyond identifying people, location, and organization to book titles, email addresses, phone numbers, and protein names (Nadeau and Sekine 2007). Recently there has been a surge of interest in extracting product attributes from online data due to the rapid growth of E-Commerce. Current work in this domain focuses on mining product reviews and descriptions from retailer websites. Such text data tend to be long and generate enough context for the target task (Brody and Elhadad 2010; Liu et al. 2005; Popescu and Etzioni 2005). In this paper, we focus on mining short product listing titles, which poses unique challenges.

Short listings are typical in classified ads where each seller is given limited space (in terms of words) to describe the product. On eBay, product listing titles cannot exceed 55 characters in length. Similarly, on Craigslist and newspaper ads, the length of a listing title is restricted. Extracting product attributes from such short titles faces the following challenges:

- Loss of grammatical structure in short listings where many nouns are piled together.
- Typographical errors, abbreviations, and acronyms that must be normalized to the standardized values.
- Lack of contextual information to infer product attribute value.

It can be argued that the use of short listings simplifies the problem of attribute extraction, since short listings can be easily annotated and one can apply supervised learning approach to extract product attributes. However, as the size of the data grows, obtaining labeled training set on the scale of millions of listings becomes very expensive. In such a scenario, incorporating unlabeled examples in a semi-supervised fashion to scale up the solution becomes a necessity rather than a luxury.

We formulate the product attribute extraction problem as a named entity recognition (NER) task and investigate supervised and semi-supervised approaches to this problem. In addition, we have investigated attribute discovery, and normalization to standardized values. We use listings from eBay's clothing and shoes categories and develop an attribute extraction system for 4 attribute types. We have $105,335$ listings from men's clothing category and $72,628$ listings from women's clothing category

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1557–1567,
Edinburgh, Scotland, UK, July 27–31, 2011.

on eBay, constituting a dataset of $1,380,337$ word tokens.

In the first part of this work, we outline a supervised learning approach to attribute value extraction where we train a sequential classifier and evaluate the extraction performance on a set of hand-labeled listings. Using maximum entropy and SVM as the base classifier (for classifying the individual word tokens), a hidden Markov model (HMM) is trained on the the probabilistic output of the base classifier, and a sequential label prediction is obtained using a Viterbi decoding. We show a performance comparison of supervised HMM, MaxEnt, SVM, and CRF for this task.

In the second part of our work, to grow our seed list of attributes, we present a bootstrapped algorithm for attribute value discovery and normalization, honing in on one particular attribute (brand). The goal is given an initial list of unambiguous brands, we grow the seed dictionary by discovering context patterns that are often associated with such attribute type. First, we automatically partition data into a training/test set by labeling word tokens in each listing using exact matching to entries in the dictionary. Brand phrases that can be confused with other attributes, e.g. the word *camel* — both a brand and a color — will not be a part of this initial seed list to create the training set. A classifier is then trained to learn context patterns surrounding the known brands from the training set, and is used to discover new brands from the test set.

Finally, for known attribute values, we normalize the results to match to words in our dictionary. Normalizing the variants of a known brand to a single normalized output value is an important aspect of a successful information extraction system. To this end, we investigate several string similarity/distance measures for this task and found that n-gram substring similarity (Kondrak 2005) yields accurate normalization results.

The main contribution of this work is a product attribute extraction system that addresses the unique problems of information extraction from short listing titles. We combine supervised NER with bootstrapping to expand the seed list, and investigate several methods to normalize the extracted results. Our system has been tested on large-scale eBay listing datasets to demonstrate its effectiveness.

2 Related Work

Recent work on product attribute extraction by (Brody and Elhadad 2010) applies a Latent Dirichlet Allocation (LDA) model to identify different aspects of products from user reviews. Similar work is presented in (Liu et al. 2005). Topic models such as LDA groups similar words together by identifying topics (product aspects) from patterns of word-occurrences. Such grouping can discover new aspects of a product such as "portability" (for netbook computers), but it may generate aspects that are vague and not easily interpretable. Indeed, how to refine discovered aspects and clean up words in each aspect remains an open question. The LDA approach also treats documents as bags of words, where important information in word sequences is not taken into account in learning the model.

Our work is most closely related to (Ghani 2006), where a set of product attributes of interests are predefined and a supervised learning method is applied to extract the correct attribute values for each class. Starting out from a small set of training examples, a bootstrapping technique is used to generate more training data from unlabeled data. The main difference to our method lies in how bootstrapping is used. (Ghani 2006) used EM to add more training data from unlabeled data, while in our approach bootstrapping is used to expand the seed list. First, we automatically generate labeled data by matching seed list to unlabeled data. Then, these auto-labeled training set is used to train a classifier to identify new attribute values from a separate set of unlabeled data. Thirdly, newly discovered product attribute values are added back to our seed list. Thus our original classifier for product attribute extraction can be improved through an expanded seed list.

In (Ghani and Jones 2002; Jones 2005), several bootstrapping methods are compared. These methods include self-training, co-EM and EM. All of these approaches are different from ours, as described in detail earlier. In (Probst et al. 2006), a Naive Bayes learner is combined with Co-EM to generate more training data from unlabeled data, and attribute-value pairs are extracted on adjacent words.

The automatic bootstrapping in this paper was inspired by (Pakhomov 2002)—an acronym expansion algorithm for medical text documents. The underlying assumption is that abbreviated forms and their

corresponding expansions occur in similar contexts; consequently, the surrounding context patterns can be used in associating the correct expansion to its acronym.

Our seed list expansion algorithm indeed bears some similarity to the work of (Nadeau el al 2006) and (Nadeau 2007). In (Nadeau el al 2006), automobile brands are learned automatically from web page context. First, a small set of 196 seed brands are extracted together with their associated web page contexts from popular news feed. The web context is subsequently used to extract additional automobile brands, which result in a total of 5701 brands. However, the reported results in (Nadeau el al 2006) have low precision, in some case less than 50%. Eventually their approach needs to rely on rule-based ambiguity resolver to increase the precision. Our system does not rely on manually created rules.

A more NLP-oriented approach is proposed in (Popescu and Etzioni 2005), where noun phrases are extracted from online user reviews. Their system tries to identify product features and user opinions from such noun phrases. A PMI (pointwise mutual information) score is evaluated between each noun phrase and discriminators associated with the product class. The noun-phrase approach does not work well in informal texts. In our case, user-generated short product listings may have many nouns concatenated together without forming a phrase or obeying correct grammatical rules.

Finally, another similar bootstrapping method is presented in (Mintz et al. 2009), where instances of known entity relations (or seed list in our paper) are matched to sentences in a set of Wikipedia articles, and a learning algorithm is trained from the surrounding features of the entities. The trained model is then applied to a test set of Wikipedia articles, and has been reported to be able to discover new instances. In our case, we apply our learned model to a new test set, and discover new brand names from the listings.

The nature of non-grammatical text we face makes our work similar to the NER work on informal texts. (Minkov et al. 2005) proposes an NER system that extracts personal names from emails. The work in (Gruhl et al 2009) identifies song titles from online forums on popular music, where song titles can be very ambiguous. By using real-world constraints such as known song titles, (Gruhl et al 2009) restricts the set of possible entities and are able to obtain reasonable recognition performance.

3 Corpus

The data used in all analysis in this paper is obtained from eBay's clothing and shoes category. Clothing and shoes have been important revenue-generating categories on the eBay site, and a successful attribute extraction system will serve as an invaluable tool for gathering important business and marketing intelligence. For these categories, the attributes that we are interested in are *brand (B)*, *garment type/style (G)*, *size (S)*, and *color (C)*. We gather $105,335$ listings from men's clothing category and $72,628$ listings from women's clothing category, constituting a dataset of $1,380,337$ word tokens. On average, each listing title contains 7.76 words.

A few examples of listings from eBay's clothing and shoes categories are shown in Fig 1. When designing an attribute extraction system to distinguish between the 4 attribute types, we must take into account the fact that individual words alone — without considering context — are ambiguous, as each word can belong to multiple attribute types. To give concrete examples, *inc* is a brand name of women's apparel but many sellers use it as an acronym for inch (brand vs. size). The word *blazer* can be a brand entity or it can be a garment type (brand vs. garment type). In addition, like other real-world user-generated texts, eBay listings are littered with site-specific acronyms, e.g. *BNWT* (brand new with tag), *NIB* (new in box), and abbreviations introduced by individual sellers, e.g. *immac* (immaculate), *trs* (trousers). In designing an information extraction system for our dataset, we need to account for the general as well as specific properties of our dataset.

4 Supervised Named Entity Recognition

In the first part of this work, we adopt a supervised named entity recognition (NER) framework for the attribute extraction problem from eBay listing titles. The goal is to correctly extract attribute values corresponding to the 4 attribute types, from each listing. One key assumption of the supervised learning paradigm is the availability of a labeled training data for training a classifier to distinguish between different classes. We generate our training data in

Figure 1: Example listings and their corresponding labels from the clothing and shoes category.

the following manner. For each listing, we remove extraneous punctuation symbols (*,(,),!,:,;) and tokenize each listing into a sequence of tokens. Given 4 dictionaries of seed values for the 4 attribute types, we match n-gram tokens to the seed values in the dictionaries, and create an initial round of labeled training set, which must then be manually inspected for correctness. In this work, we tagged and manually verified $1,000$ listings randomly sampled from the $105,335$ listings from the men's clothing category, resulting in a total of $7,921$ labeled tokens with $1,521$-word vocabulary. Fig. 1 shows examples of labeled listings, with tags *B* corresponding to brand, *C* for color, *S* for size, *G* for garment type/style, and *NA* for none of the above.

4.1 Classifiers

One of the most popular generative model based classifiers for named entity recognition tasks is Hidden Markov Model (HMM), which explicitly captures temporal statistics in the data by modeling state (label/tag) transitions over time. Discriminative classifiers, which directly model the posterior distribution of class label given features, i.e. SVM (Isozaki and Kazawa 2002) and Maximum Entropy model for NER (Chieu and Ng 2003), have been shown to outperform generative model based classifiers. More recently, Conditional Random Fields (CRF) (Feng and McCallum 2004; McCallum 2003) has been proposed for a sequence labeling problem and has been established by many as the state-of-the-art model for supervised named entity recognition task. In this section, we briefly summarize the pros and cons of each approach.

4.1.1 Hidden Markov Models

A hidden Markov model (HMM) is a probabilistic generative model for sequential data. HMM is characterized by 2 sets of model parameters — emission probabilities which produce the observation variable given the hidden state, and the state transition probability matrix which captures the temporal correlation in the hidden state sequences. Given a set of labeled training sequences as shown in Figure 1, one can train an HMM to model temporal statistics in the observation sequences. In our task, a sequence of word tokens from listing titles are our observations. One simple approach to use HMM is to set a hidden state to correspond to a tag class. In the training phase, since all the tags are given, the hidden states indeed become visible and inference in this model becomes much more simplified. The multinomial parameter for the emission probabilities $p(w|s)$ can be learned with a closed-form update (maximum likelihood estimate). During testing, however, an efficient forward-backward algorithm must be used to infer the most likely tag sequence that accounts for the observation.

One main drawback of HMM is the type of features that it can handle. Like other probabilistic generative models, in order to account for rich, overlapping feature sets, e.g. text formatting features, the correlation structures in the overlapping features must be explicitly modeled. Indeed, in the classic HMM based NER, the simple feature used is the word identity itself, which might not be sufficiently discriminative in distinguishing between different classes. In addition, because of data sparsity (out-of-vocabulary) problem due to the long-tailed distribution of words in natural language, sophisticated unknown word models are generally needed for good performance (Klein et al. 2003).

4.1.2 Maximum Entropy models

The principle of maximum entropy states that among all the distributions that satisfy feature constraints, we should pick the distribution with the highest entropy, since it makes the least assumption about the data and will have better generalization capability to unseen data. Maximum entropy classifier, therefore, is the highest entropy conditional distribution of the class label given features, which has been shown to conveniently take an exponential form. Maximum entropy classifier is thus closely related to logistic regression model.

Position Features:
- Position from the beginning of listing
- Position to the end of listing
Orthographic Features:
- Identity of the current word
- Current word contains a digit
- Current word contains only digits
- Current word is capitalized
- Current word begins with a capitalized letter followed by all non-cap letters.
- Current word is &
- Current word is £
- N-gram substring features of current word ($N = 4, 5, 6$)
Context Features:
- Identity of 2 words before the current word
- Identity of 2 words after the current word
- Previous word is *from*
- Previous word is *by*
- Previous word is *and*
- N-gram substring features of neighboring words ($N = 4, 5, 6$)
Dictionary Features:
- Membership to the 4 dictionaries of attributes
- Exclusive membership to dictionary of brand names
- Exclusive membership to dictionary of garment types
- Exclusive membership to dictionary of sizes
- Exclusive membership to dictionary of colors

Table 1: Feature set used in discriminative classifiers.

MaxEnt classifiers (Ratnaparkhi 1996; Ratnaparkhi 1998) have been applied to various NLP applications. The attraction of the framework lies in the ease with which different information sources used in the modeling process are combined and the good results that are reported with the use of these models. The set of redundant features used for the MaxEnt classifier is the same as those used for the SVM classifier, which we outline in the next section.

4.1.3 Support Vector Machines

Support Vector Machine (SVM) is yet another popular classifier for a supervised NER task. In a binary classification case, SVM finds parameters of a linearly separating hyperplane that best separates data from the 2 classes, in a sense that the margin of separation is maximized. Since only the samples closest to the decision boundary (the so-called support vectors) determine the location of the separating hyperplane, SVM can be trained on very few training examples even for data in a high-dimensional space. For our supervised NER system, we use the following features, as described in detail in Table 1, as input to the discriminative classifiers.

The use of char N-gram (N-gram substring) features was inspired by the work of (Klein et al. 2003), where the introduction of such features has been shown to improve the overall F1 score by over 20%. In (Kanaris et al. 2006), char N-gram features consistently outperform word features in learning effective spam classifiers. Indeed the use of character N-gram features as an input to the classifier subsumes the use of prefix, suffix, and the entire word features. Generally speaking, char N-gram features provide a more robust representation against misspelling since string $s1$ and its spelling variant $s2$ may share many char N-gram substrings in common.

POS and punctuation features are not used in our NER system. This is mainly due to the fact that eBay listing titles are not complete sentences and the output from running a POS tagger through such data can indeed be unreliable. For punctuation features, eBay sellers are known to abuse punctuation marks excessively to draw attention of the potential buyers to click on their listings. In addition, we find that morphological features are less predictive of entity names in eBay listing titles than they are in formal documents. To give a concrete example, capitalization is a good predictor of entity names in traditional NER systems, but on the eBay site, many sellers use all-cap or all-lowercase letters for every word in their titles, bringing into question the discriminative power of widely used features in traditional NER systems.

4.1.4 Viterbi Smoothing

The Viterbi algorithm can be used to smooth the prediction output from SVM or MaxEnt. More specifically, the Viterbi decoder enforces the temporal consistency on the individual label prediction as inferred by the base classifier — MaxEnt or SVM, independently based on the feature representation of each word token. The probabilistic ouput of the base classifier is the observation or evidence, while the temporal consistency is encoded in the empirical state transition probability matrix inferred from the training data. This scenario is analogous to comparing MAP (maximum a priori) estimate with that of ML (maximum likelihood) in that the former incorporates a prior belief when making a final estimate of the parameter values (most likely label sequence predicted by the Viterbi algorithm), while the latter uses only the observation to infer the most likely parameter estimate (independently inferred predicted labels of each word token from the base classifier).

We adopt the approach from the work of (Chieu and Ng 2003), which uses Viterbi to improve the classification results from MaxEnt classifier for NER tasks. Instead of computing the transition probability matrix by recording the frequency of how many times state i at time T transitions to state j at time $T+1$, we simply record that this state i to j transition is admissible. This approach, indeed, divides a set of all label sequences into ones that are admissible and inadmissible, and assign equal probabilities to all the admissible sequences. Such an approach therefore eliminates all the inadmissible sequences of labels (i.e. prohibit the scenario where *-in* sub-tag is followed by *-begin* sub-tag), while allowing the Viterbi algorithm to give more weight to the classification outputs from SVM or MaxEnt.

4.1.5 Conditional Random Field (CRF)

Conditional Random Field, since its conception in the seminal work of (Lafferty et al. 2002), is a discriminative classifier for sequential data that combines the best of both worlds. Like SVM and MaxEnt, CRF is a discriminative classifier that directly models the conditional distribution of the target variable given the observed variable, i.e. no modeling resource is wasted in modeling complex correlation structures in the observation sequences. Like HMM, CRF makes prediction on the label sequence by incorporating the temporal smoothness. Indeed CRF has been established by many as the state-of-the-art supervised named entity recognition system for traditional NER tasks (Feng and McCallum 2004; McCallum 2003), for NER in biomedical texts (Settles 2004), and in various languages besides English, such as Bengali (Ekbal et al. 2008) and Chinese (Mao et al 2008). Various modifications to CRF have recently been introduced to take into account of non-local dependencies (Krishnan and Manning 2006) or broader context beyond training data (Du et al. 2010).

4.2 Experimental Results

In this section, we compare the generative model based and discriminative model classifiers for supervised NER tasks. Given $1,000$ manually tagged listings from the clothing and shoes category in eBay, we adopt a 90-10 split and use 90% of the data for training and 10% for testing. Each listing title is tokenized into a sequence of word tokens, each manually assigned to one of the 5 tags: brand (*B*), size (*S*), color (*C*), garment type (*G*), and none of the above (*NA*). In order to more accurately capture the boundary of multi-token attribute values, we further sub-divide each tag into 2 classes using *-beg* and *-in* sub-tags. This step increases the number of classes that our classifier needs to handle from 5 to 9 classes given as follows: {*B-beg*, *B-in*, *C-beg*, *C-in*, *S-beg*, *S-in*, *G-beg*, *G-in*, and *NA*}.

	SVM	**MaxEnt**	**HMM**	**CRF**
w/o Viterbi	89.05%	87.64%	-	-
w/ Viterbi	**89.47%**	**88.13%**	**83.82**	**93.35%**

Table 2: Classification accuracy (%) on 9-class NER on men's clothing dataset, comparing SVM, MaxEnt, supervised HMM, and CRF.

Table 2 shows a comparison of classification accuracy from 4 classifiers — SVM, MaxEnt, HMM, and CRF. Supervised HMM, with the most simplistic feature, yields the baseline result at 83.82% accuracy. All the discriminative classifiers — CRF, MaxEnt, and SVM — outperform the baseline by HMM, with CRF improving on the baseline performance by the largest margin, concurring to other reports of its state-of-the-art results. Indeed, when using exactly the same set of features as SVM and MaxEnt, the performance of CRF indeed drops to 89.11%, which is on par with that of SVM and MaxEnt. However, when restricting to using dictionary and word identity features, the performance of CRF improves, indicating the importance of feature selection to such model. SVM and MaxEnt yield similar performance with SVM slightly outperforming MaxEnt classifier by 1.6%. The incorporation of temporal smoothness constraint enforced by the Viterbi algorithm slightly improves the label sequence prediction (comparing row 1 and row 2 in Table 2).

The HMM implementation used in our experiments is the Hunpos tagger in (Halacsy et al. 2007), which captures the state transitional probabilities using second-order Markov model. For SVM, we use the popular libSVM package (Chang and Lin 2001) which produces probabilistic output from fitting a sigmoid function to the distances between samples and the separating hyperplane. We use linear kernel in our experiments, although RBF kernel with grid search for optimal parameters yield slightly superior

performance, with a significantly higher computational cost. The MaxEnt implementation used in our experiment is the version available from the NLTK toolkit, with BFGS optimizer. For CRF, we use the linear-chain CRF model available from the Mallet package[1].

5 Bootstrapping for Dictionary Expansion

The supervised learning approach assumes the existence of an annotated set of training data. Often times, training data must be painstakingly marked up and collecting large-scale labeled training examples can be very costly. In recent years, more and more research effort has been focused on how to leverage a vast amount of unlabeled data in a semi-supervised or entirely unsupervised fashion for NER as well as for other similar NLP tasks, e.g. POS tagging, sentence boundary detection, and word sense disambiguation (Riloff 1999; Ghani and Jones 2002; Probst et al. 2006; Brody and Elhadad 2010; Haghighi 2010).

One way to incorporate a vast amount of unlabeled data is to learn a clustering of words that assigns syntactically similar words to the same clusters. Popular clustering algorithms used prevalently in many NER systems are, for example, the combination of distributional and morphological similarity work of (Clark 2003) or the classic N-gram language model based clustering algorithm of (Brown et al. 1992). In such a system, when training an NER classifier, we introduce a word cluster id as an additional feature in the input, with the hope that the model will pick out clusters that are highly indicative of each class. When encountering words that are out-of-vocabulary (OOV) in the test set, if those words are assigned the same cluster membership as some other words in the training set, the cluster feature will fire, allowing for correct classification results to be obtained (Lin and Wu 2009; Faruqui and Pado 2010).

5.1 Growing Seed Dictionary

In this work, we focus on the problem of how to grow the seed dictionary and discovering new brand names from eBay listing data. While the performances of supervised NER classifiers as described in sections 4.1.1-4.1.5 are satisfactory, in practice, however, especially with a small training set size, we often find that the trained model puts too much weight on the dictionary membership feature and new attribute values are not properly detected. In this section, instead of using the seed list of known attribute values as a feature into a classifier, we use the seed values to automatically generate labeled training data. For the specific case of brand discovery, this initial list used to generate training data must contain only names that are unambiguously brands. We hence remove ambiguous names or phrases that belong to multiple attribute types from the list, such as *jumpers*(both a brand name and a garment type), or (ii) *camel* is a short name of brand *Camel active* as well as a color, or (iii) *lrg* is an acronym for a brand as well as an acronym for large which specifies size.

The training/test data is generated by matching N-gram tokens in listing titles to all the entries in the initial brand seed dictionary. Following the convention in (Minkov et al. 2005), we use the following set of 5 tags, (1) one-token entity (B1 tag) (2) first token of a multi-token entity (Bo tag for Brand-open) (3) last token of a multi-token entity (Bc tag for Brand-close) (4) middle token of a multi-token entity (Bi tag for Brand-inside) (5) token that is not part of a brand entity (NA tag). The listings with at least one non-NA tags are put in the training set, and listings that contain only NA tags are in the test set. Similar to the acronym expansion algorithm of (Pakhomov 2002) which learns contexts that associate acronyms to their correct expansions, the intuition behind our work in this section is that the classifier, trained on a labeled training set of known brands, learns context patterns that can discriminate the current word as being a brand (more precisely as part of a brand) from the other attribute types, which are now lumped together as *NA*.

5.2 Experiments

In the first experiment, a set of $72,628$ listings from the women's clothing category is partitioned into a training set of $39,448$ listings and test set of $33,180$ listings based on an initial seed list of known $6,312$ women's apparel brands manually prepared by our fashion experts. The partitioning is done, as described in great detail above, in such a way that known brands in the seed list do not exist in the

[1] http://mallet.cs.umass.edu/

Women's Clothing	Men's Clothing	Garment Type
'monsoon'	henley's	nightshirt
riverislandtop	abercrombie&fitch	cargoshorts
dorothyperkins	lacost	trenchcoat
river islanfd	versace	sweatpants
marks&spencers	sonnetti	cardigans
river islands	supremebeing	boardshorts
river islan	brookhaven	tracksuite
monsoomn	guiness	swimshorts
dorothry perkins	'next'	trouses
principle	suprerdry	microfleece
?river island	henbury	boilersuit
bnwtmonsoon	paul smiths	snopants
marella	ricci	pjs
soulcal	craghopper	jkt

Table 3: Discovered attribute values, ranked order by their confidence scores. (Left) Discovered brands from Women's clothing category. We use 6,312 brands as seed values. (Middle) Discovered brands from Men's clothing category, with 3,499 seed values used. (Right) Discovered garment types (styles) from Men's clothing category, learned from 203 seed values.

test data (using exact string matching criterion). We train a 5-class MaxEnt classifier and adopt the same feature sets as described in Section 4.1.3. During the test phase, the classifier predicts the most likely brand attribute from each listing, where we are only interested in the predictions with confidence scores exceeding a set threshold. We ranked order the predicted brands by their confidence scores (probabilities) and the top 300 unique brands are selected. We manually verify the 300 predicted brands and found that 90.33% of the predicted brands are indeed names of designers or women's apparel stores (true positive), resulting a precision score of 90.33%.

Indeed, the precision score presented above is obtained using an exact matching criterion where partial extraction of a brand is regarded as a miss, i.e. our extractor extracts only *Calvin* when *Calvin Klein* is present in the listing (false positive). The left column of Table 3 shows examples of newly discovered brands from Women's clothing category. Many of these newly discovered brands are indeed misspelled versions of the known brands in the seed dictionary.

The middle column of Table 3 shows a set of Men's clothing brands learned automatically from a similar experiment conducted on a set of $105,335$ listings from Men's clothing category. Using an initial set of $3,499$ known brand seeds, we partition the dataset into a training set of $67,307$ listings and a test set of $38,028$ listings (for later reference we refer to this test set as set A). Based on the top 200 predicted brands, 179 of which are verified as being true positive samples, resulting in 89.5% precision. We carry out a similar experiment to grow the seed dictionary for garment type, and are able to identify the top 60 new garment types. 54 out of 60 are true positive samples, resulting in precision score = 90%. Examples of the newly discovered garment types are shown in Table 3 (right column), where abbreviated forms of garment types such as *jkt* (short for jacket) and *pjs* (short for pajamas) are also discovered through our algorithm.

Seed list	Test set 1	Test set 2
Orig. seeds	83.56%	90.02%
Orig. seed + 200 new brands	92.75%	93.66%

Table 4: NER Accuracy on 2 test sets as the seed dictionary for brands grows. Results shown here are obtained the same Men's clothing category dataset, as used to show the supervised NER results in Table 2.

By adding these newly discovered attributes back to the dictionary, we can now re-evaluate our supervised NER system from section 4 with the grown seed list. To this end, we construct 2 test sets from the same $105,335$ listings of Men's clothing category as used in Section 4. Test set 1 is a set of 500 listings randomly sampled from the $38,028$-listing subset known not to contain any brands in the original brand seed dictionary (set A). As seen in Table 4, an improvement of 9% in accuracy results from the use of the grown seed list. Since this dataset is known to not contain any brands from the original brand seed dictionary, the addition of 200 new brands solely accounts for all the accuracy boost. Test set 2 is constructed slightly differently by randomly sampling 500 listings from the entire $105,335$ listings of Men's clothing category. As seen in Table 4, a smaller improvement of 3.7% is observed.

6 Normalization

With the above described brand discovery algorithm, the newly discovered brands from the test set can be grouped into 2 categories — (i) misspelling, spelling

invariants, abbreviated forms of known brands in the seed list or (ii) novel brands or clothing/shoes designers, which are not members of the original seed list. Normalizing the variants of a known brand to a single normalized output value is an important aspect of our attribute extraction algorithm, as these variants account for over 20% of listings in the eBay clothing and shoes category. When gathering business/marketing intelligence, missing out on 20% of the data could skew the calculation of supply, demand, and pricing metrics, and eventually lead to the wrong policy decision made.

The problem of alternate spellings of names has been addressed in the database community successfully using fuzzy string matching algorithms e.g. Soundex or string edit distance. In this work, since the attribute values are often partially extracted, i.e. a word in a multi-word phrase is extracted, in order to match to the correct normalized value, we must investigate robust substring matching algorithms suitable for partial matching. To this end, we explore 2 string similarity/distance measures for normalizing the extracted attributes. First, we investigate n-gram similarity measures defined as the number of shared character n-grams, i.e. substrings of length n (Kondrak 2005). More specifically, a string similarity measure between $s1$ and $s2$ is defined as the percentage of common substrings of length n (out of all substrings of length n). This similarity measure is quite robust to partial matching, as a two-word phrase can appear out of order while most of the character n-grams, where n = 3, remain virtually unchanged. Certainly, finding the right value of n will greatly impact the matching performance of the algorithm. In our experiment, we find the optimal n for brands to be 3 and 4. Table 5 shows a few examples of normalized outputs as a result of finding the best match for the extracted brand names from among a set of predefined normalized values. When the best matching score falls below a threshold, we declare no match is found and classify the extracted brand as a new brand.

Another distance measure that we explore is the Jaro-Winkler distance. Designed to be more suitable for matching short strings such as people's names, Jaro-Winkler distance is defined based on the number of character transpositions and the number of matching characters. In addition, a prefix scale p parameter is used and can be tuned to weigh more favorably on strings that match from the beginning for a set prefix length. In our experiments with brand normalization, over 50% of the matches from the Jaro-Winkler distance are, however, identified as being incorrect.

Extracted brands	Normalized values
river islands	river island
fruit of loom	fruit of the loom
fruit loom	fruit of the loom
'ralph lauren	ralph lauren
mark & spencer	marks & spencer
yvessaintlaurent	yves saint laurent
yves st laurent	yves saint laurent
combats	combat
'kickers'	kickers
kickers	kickers
armarni	armani
abrecrombie	abercrombie
life & limb	NEW BRAND
oliver baker	NEW BRAND
haines & bonner	NEW BRAND
dehavilland	NEW BRAND
nigel cabourn	NEW BRAND

Table 5: Extracted brands and their corresponding normalized values.

7 Conclusion

In this work, we have described an information extraction system for applications in the domain of inventory/business Intelligence. The goal is given an eBay listing title, our system correctly extracts the defining attributes in order to associate each item to a specific product. We investigate and compare several supervised NER systems — supervised HMM, SVM, MaxEnt, and CRF — and found SVM and MaxEnt with Viterbi decoding to yield the best performance. Focusing on the clothing and shoes categories on eBay's site, we presented a bootstrapped algorithm that can identify new brand names corresponding to (1) spelling invariants or typographical errors of the known brands in the seed list and (2) novel brands or designers. Our attribute extractor correctly discovers new brands with over 90% precision on multiple corpora of listings. To output normalized attribute values, we explore several fuzzy string comparison algorithms and found n-gram substring matching to work well in practice.

8 Acknowledgment

The authors would like to thank Nalini Johnas and Padmanaban Ramasamy for their help in gathering listing data used in all of our experiments.

References

A. Berger, S. Pietra, V. Pietra, *A Maximum Entropy Approach to Natural Language Processing*, ACL 1996.

S. Brody, N. Elhadad, *An Unsupervised Aspect-Sentiment Model for Online Reviews*, HLT-NAACL 2010.

P. Brown, P. deSouza, R. Mercer, V. Della Pietra, J. Lai, Class-based n-gram Models of Natural Language, ACL 1992.

C.-C Chang, C.-J. Lin, LibSVM: A Library for Support Vector Machines (2001).

H. L. Chieu, H. T. Ng, *Named Entity Recognition with a Maximum Entropy Approach*, ACL 2003.

A. Clark, Combining Distributional and Morphological Information for Part of Speech Induction, EACL 2003

G. Demartini, C. S. Firan, M. Georgescu, T. Iofciu, R. Krestel, and W. Nejdl, *An Architecture for Finding Entities on the web*, Latin American Web Congress 2009.

J. Du, Z. Zhang, J. Yan, Y. Cui, and Z. Chen. *Using search session context for named entity recognition in query*. In SIGIR10, Geneva, Switzerland, July 19-23 2010.

Asif Ekbal, Rejwanul Haque, and Sivaji Bandyopadhyay. 2008. *Named entity recognition in Bengali: A conditional random field approach*. In Proceedings of IJCNLP, pages 589594.

M. Faruqui, S. Pado, Training and Evaluating a German Named Entity Recognizer with Semantic Generalization, Proceedings of Konvens 2010, Saarbrucken, Germany.

F. Feng, A. McCallum, *Chinese segmentation and new word detection using conditional random fields*, in COLING 2004.

J. R. Finkel, T. Grenager, and C. Manning, *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling*, ACL 2005.

J. R. Finkel, C. Manning, *Nested Named Entity Recognition*, EMNLP 2009.

R. Ghani, K. Probst, Y. Liu, M. Krema, A. Fano, *Text Mining for Product Attribute Extraction*, SIGKDD, 2006.

R. Ghani, R. Jones, *A comparison of efficacy and assumptions of bootstrapping algorithms for training information extraction systems*, Workshop on Linguistic Knowledge Acquisition and Representation at the Third International Conference on Language Resources and Evaluation (LREC), 2002.

T. Grenager, D. Klein, and C. D. Manning, *Unsupervised Learning of Field Segmentation Models for Information Extraction*, ACL 2005.

D. Gruhl, M. Nagarajan, J. Pieper, C. Robson, and A. Sheth. *Context and Domain Knowledge Enhanced Entity Spotting In Informal Text*. In Proceedings of the 8th International Semantic Web Conference (ISWC 2009). Springer, 2009.

A. D. Haghighi, *Unsupervised Models of Entity Reference Resolution*, Ph. D. Thesis, University of Calfornia, Berkeley, 2010.

P. Halacsy, A. Kornai, C. Oravecz, *HunPos: an open source trigram tagger*, ACL 2007.

H. Isozaki and H. Kazawa, *Efficient Support Vector Classifiers for Named Entity Recognition*, ACL 2002.

R. Jones, *Learning to Extract Entities from Labeled and Unlabeled Text*, PhD Thesis, 2005.

I. Kanaris, K. Kanaris, I. Houvardas, E. Stamatatos, *Words vs. Character N-grams for Anti-spam Filtering*, International Journal on Artificial Intelligence Tools, 2006.

D. Klein, J. Smarr, H. Nguyen, C. Manning, *Named Entity Recognition with Character-level Models*, CoNLL 2003.

R. Koeling, *Chunking with Maximum Entropy Models*, Proc. of CoNLL-2000.

G. Kondrak, *N-Gram Similarity and Distance*, SPIRE 2005.

V. Krishnan and C. D. Manning, *An effective two-stage model for exploiting non-local dependencies in named entity recognition*, in ACL-COLING, 2006.

T. Kudo, Y. Matsumoto, *Chunking with Support Vector Machines*, ACL 2001.

J. Lafferty, A. McCallum, F. Pereira, *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*, ICML 2002.

V. I. Levenshtein, *Binary code capable of correcting deletions, insertions, and reversals.* Phs. Dokl., 6:707-710.

D. Lin, X. Wu, *Phrase Clustering for Discriminative Learning*, ACL 2009.

B. Liu, M. Hu, and J. Cheng, *Opinion Observer: Analyzing and Comparing Opinions on the Web*, WWW 2005.

Xinnian Mao, Saike He, Sencheng Bao, Yuan Dong, and Haila Wang, *Chinese Word Segmentation and Named Entity Recognition Based on Conditional Random Fields*, Sixth SIGHAN Workshop on Chinese Language Processing, 2008

A. McCallum, *Efficiently Inducing Features of Conditional Random Fields*, UAI 2003.

A. McCallum, D. Jensen, *A Note on Unification of Information Extraction and Data Mining using Conditional-Probability, Relational Models*, Proceedings of IJCAI-2003 on Learning Statistical Models from Relational Data, 2003.

J. F. McCarthy, *A Trainable Approach to Coreference Resolution for Information Extraction*, Ph. D. Thesis, University of Massachusetts at Amherst, 1996.

E. Minkov, R. C. Wang, and W. W. Cohen, *Extracting Personal Names from Email: Applying Named Entity Recognition to Informal Text*, ACL 2005.

Mike Mintz, Steven Bills, Rion Snow, Daniel Jurafsky. 2009. *Distant Supervision for Relation Extraction without Labeled Data*, In Proceedings of ACL/AFNLP 2009.

S. Moghaddam, M. Ester, *Opinion Digger: An Unsupervised Opinion Miner from Unstructured Product Reviews*, CIKM 2010

David Nadeau, P. Turney, S.Matwin, *Unsupervised Named Entity Recognition: Generating Gazetteers and Resolving Ambiguity*. In Proc. Canadian Conference on Artificial Intelligence, 2006.

David Nadeau and Satoshi Sekine. *A survey of named entity recognition and classification*. Linguisticae Investigationes, 30(1):326, 2007.

Nadeau, D., *Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision*, PhD thesis, University of Ottawa, 2007.

S. Pakhomov, *Semi-supervised Maximum Entropy Based Approach to Acronym and Abbreviation Normalization in Medical Texts*, ACL 2002.

A.-M. Popescu, O. Etzioni, *Extracting Product Features and Opinions from Reviews*, EMNLP 2005.

K. Probst, R. Ghani, M. Krema, A. Fano, *Semi-Supervised Learning to Extract Attribute-Value Pairs from Product Descriptions on the Web*, ECML 2006.

V. Punyakanok, D. Roth, *The use of classifiers in sequential inference*, NIPS 2001.

H. Raghavan, J. Allan, Matching Inconsistently Spelled Names in Automatic Speech Recognizer Output for Information Retrieval, HLT-EMNLP 2005.

A. Ratnaparkhi, *A Maximum Entropy Part of Speech Tagger.* In EMNLP 1996.

A. Ratnaparkhi, *Maximum Entropy Models forNatural Language Ambiguity Resolution*, Ph. D. Thesis, University of Pennsylvania.

E. Riloff, R. Jones, *Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping*, AAAI 1999.

Settles, B. (2004), *Biomedical named entity recognition using conditional random fields and rich feature sets*, in Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA), 2004, Geneva, Switzerland.

W. M. Soon, H. T. Ng, D. Chung, Y. Lim, *A machine learning approach to coreference resolution of noun phrases*, Computational Linguistics, 27(4): 521-544, 2001.

H. Wallach, *Efficient Training of Conditional Random Fields*, M. Sc. Thesis, Division of Informatics, University of Edinburgh, 2002.

D. Wu, W. S. Lee, N. Ye, and H. L. Chieu, *Domain adaptive bootstrapping for named entity recognition*, EMNLP 2009.

Y. Zhao, B. Qin, S. Hu, T. Liu, *Generalizing Syntactic Structures for Product Attribute Candidate Extraction*, ACL 2010

Twitter Catches The Flu: Detecting Influenza Epidemics using Twitter

Eiji ARAMAKI
The University of Tokyo
JST PRESTO
Tokyo, Japan
eiji.aramaki@gmail.com

Sachiko MASKAWA
The University of Tokyo
Tokyo, Japan
sachiko.maskawa@gmail.com

Mizuki MORITA
National Institute of
Biomedical Innovation
Osaka, Japan
morita.mizuki@gmail.com

Abstract

With the recent rise in popularity and scale of social media, a growing need exists for systems that can extract useful information from huge amounts of data. We address the issue of detecting influenza epidemics. First, the proposed system extracts influenza related tweets using Twitter API. Then, only tweets that mention actual influenza patients are extracted by the support vector machine (SVM) based classifier. The experiment results demonstrate the feasibility of the proposed approach (0.89 correlation to the gold standard). Especially at the outbreak and early spread (early epidemic stage), the proposed method shows high correlation (0.97 correlation), which outperforms the state-of-the-art methods. This paper describes that Twitter texts reflect the real world, and that NLP techniques can be applied to extract only tweets that contain useful information.

1 Introduction

Twitter[1], a popular micro-blogging service, has received much attention recently. It is an online network used by millions of people around the world to stay connected to their friends, family members, and co-workers through their computers and mobile telephones (Milstein et al., 2010).

Nowadays, Twitter users have increased rapidly. Its community estimated as 120 million worldwide, posts more than 5.5 million messages (*tweets*) every day (reported by Twitter.com in March 2011). Twitter can potentially serve as a valuable information resource for various applications. Huberman et al. (2009) analyzed the relations among friends. Boyd et al. (2010) investigated commutation activity. Sakaki et al. (2010) addressed the detection of earthquakes. Among the numerous potential applications, this study addresses the issue of detecting influenza epidemics, which presents two outstanding advantages over current methods.

- **Large Scale**: More than a thousand messages include the word "*influenza*" each day (Nov. 2008 – Oct. 2009). Such a huge data volume dwarfs traditional surveillance resources.

- **Real-time**: Twitter enables real-time and direct surveillance. This characteristic is extremely suitable for influenza epidemic detection because early stage detection is important for influenza warnings.

Although Twitter based influenza warnings potentially offer the advantages noted above, it might also expose inaccurate or biased information from tweets like the following (brackets **[]** indicate the comments):

- *Headache? You might have **flu.*** **[Suspicions]**
- *The World Health Organization reports the avian influenza, or bird **flu**, epidemic has spread to nine Asian countries in the past few weeks.* **[General News]**

[1] http://twitter.com/

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1568–1576,
Edinburgh, Scotland, UK, July 27–31, 2011.

- *Are you coming down with **influenza**?* **[Question]**

Although these tweets include mention of "*influenza*" or "*flu*", they do not indicate that an influenza patient is present nearby. We regard such messages (merely suspicions/questions, general news, etc.) as **negative influenza tweets**. We call others **positive influenza tweets**. In our experiments, 42% of all tweets that include "*influenza*" are negative influenza tweets. The huge volume of such negative tweets biases the results.

This paper presents a proposal of a machine-learning based classifier to filter out negative influenza tweets. First, we build an annotated corpus of pairs of a tweet and positive/negative labels. Then, a support vector machine (SVM) (Cortes and Vapnik, 1995) based sentence classifier extracts only positive influenza tweets from tweets. In the experiments, the results demonstrated the high correlation (0.89 of the correlation), which is equal performance to that of the state-of-the-art method.

The specified research point of this study is two-fold:

(1) This report describes that an SVM-based classifier can filter out the negative influenza tweets (*f*-measure=0.76).
(2) Experiments empirically demonstrate that the proposed method detects the influenza epidemics with high accuracy (correlation ratio=0.89): it outperforms the state-of-the-art method.

2 Influenza Epidemic Detection

The detection of influenza epidemics is a national mission in every country for two reasons.

(1) Anti-influenza drugs, which differ among influenza types, must be prepared before the epidemics.
(2) We can only slightly predict what type of influenza will spread in any given season.

This situation naturally demands the early detection of influenza epidemics. This section presents a description of previous methods of influenza epidemic detection.

2.1 Traditional Approaches

Most countries have their own influenza surveillance organization/center: the U.S. has the Centers for Disease Control and Prevention (CDC)[2], the E.U. has its European Influenza Surveillance Scheme (EISS), and Japan has its Infection Disease Surveillance Center (IDSC). Their surveillance systems fundamentally rely on both virology and clinical data. For example, the IDSC gathers influenza patient data from 5,000 clinics and releases summary reports. Such manual systems typically have a 1–2 week reporting lag. This time lag is sometimes pointed out as a major flaw.

2.2 Recent Approaches

In an attempt to provide earlier influenza detection, various new approaches are proposed each year.

Espino et al. (2003) described a telephone triage service, a public service, to give advice to users via telephone. They investigated the number of telephone calls and reported a significant correlation with influenza epidemics.

Magruder (2003) used the amount of over-the-counter drug sales. Because an influenza patient usually requires anti-influenza drugs, this approach is reasonable. However, in most countries, anti-influenza drugs are not available at the drug store (only hospitals provide such drugs).

The state-of-the-art approach is that proposed by Ginsberg et al. (2009). They used Google web search queries that correlate with an influenza epidemic. Their approach demonstrated high accuracy (average correlation ratio of 0.97; min=0.92; max=0.99)[3]. Several research groups have used similar approaches. Polgreen et al. (2008) used a Yahoo! query log. Hulth et al. (2009) used a query log of a Switzerland web search engine.

Although the above approaches use different information, they share the same approach, which is to observe patient actions directly. This approach was sufficient to obtain more numerous data than traditional services. Nevertheless, such information is unfortunately limited only to the service provider. For example, web search queries are available only for several companies: Google, Yahoo!, and Microsoft.

This paper examines Twitter data, which are widely available. Note that Paul and Dredze (2011) also propose a similar Twitter based approach. While they focus on a word distribution, this paper

[2] http://www.cdc.gov/flu/weekly/
[3] Their service is available at http://www.google.org/flutrends/ (Google Flu Trend).

employs a sentence classification (discrimination of negative influenza tweets).

3 Influenza Corpus

As described in Section 1, it is necessary to filter out **negative influenza tweets** to infer precise amounts of influenza epidemics. To do so, we constructed the influenza corpus (Section 3). Then, we trained the SVM-based classifier using the corpus (Section 4).

The corpus comprises pairs of sentences and a label (positive or negative). Several examples are presented in Table 1. This corpus was built using the following procedure.

3.1 Influenza Tweet

First, we collected 300 million tweets, starting from 2008 November to 2010 June, via Twitter API. Crawling results are presented in Figure 1. We extracted only influenza-related tweets using a simple word look-up of "*influenza*". This operation gave us 0.4 million tweets. We separated the data into two data groups.

Training Data are 5,000 tweets sent in November 2008. These were annotated by human annotators, and were then used for training.

Test Data are the other data. They were used in experiments of influenza epidemics detection. Because of the three dropout periods (Figure 1), the test data were separated into four periods (winter 2008, summer 2009, winter 2009, and summer 2010).

3.2 Positive–negative Annotation

To each tweet in the training dataset, a human annotator assigned one of two labels: positive or negative. In this labeling procedure, we regarded a tweet that meets the following two conditions as positive data.

Condition 1 (A Tweet person or Surrounding persons have Flu): one or more people who have influenza should exist *around* the tweet person. Here, we regard "*around*" as a distance in the same city. In cases in which the distance is unknown, we regard it as negative. Because of this annotation policy, the re-tweet type message is negative.

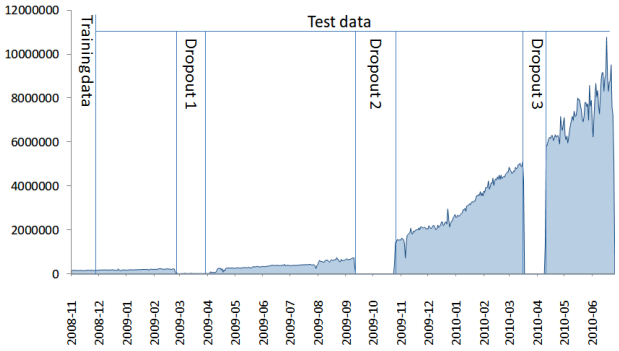


Figure 1: Twitter Data used in this Study.
The data include three dropout periods because the Twitter API specifications changed in those periods. The dropout periods were removed from evaluation in the experiments (Section 5).

Table 1: Corpus (Tweets with a Positive or Negative Label)

Positive(+1)/ Negative(-1)	Tweet
+1	A bad influenza is going around in our lab.
+1	I caught the flu. I was burning up.
+1	I think I'm coming down with the flu.
+1	It's the flu season. I had it and now he do es.
+1	Don't give me the flu. (Nearby people have the flu)
+1	My flu is worse than it was yesterday.
-1	In the normal flu season, 80 percent of deaths occur in people over 65 (Simply a fact)
-1	Influenza is now raging throughout Japan. (Too general.)
-1	His wife also contracted the bird flu, but has recovered. (Where is his wife?)
-1	You might have the flu. Has anyone around you had it? (Where are you?)
-1	Bird flu damage is spreading in Japan. (Too general.)

"+1" indicates a positive influenza tweet. "-1" indicates a negative influenza tweet. The case arc "()" indicates the reason for the positive or negative annotation.

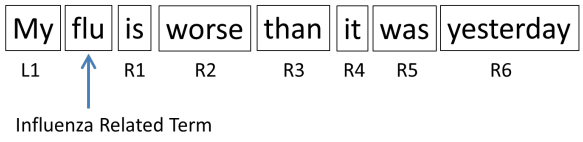


Figure 2: Feature Representation.
The word boundary is detected by a morph analyzer JUMAN[4].

[4] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html

Condition 2 (Tense/Modality): The tense should be the present tense (current) or *recent past*. Here, we define the "*recent past*" as the prior 24 hour period (such as "*yesterday*"). The sentence should be affirmative (not interrogative and not subjunctive).

4 Influenza Positive–negative Classifier

Using the corpus (Section 3), we built a classifier that judges whether a given tweet is positive or negative. This task setting is similar to a sentence classification (such as spam e-mail filtering, sentiment analysis, and so on). We used a popular means for sentence classification, which is based on a machine learning classifier under the bag-of-words (BOW) representation (Figure 2). The parameters were investigated in preliminary experiments in terms of feature window size (Section 4.1) and machine-learning methods (Section 4.2). These preliminary experiments were conducted under the ten-fold cross variation manner using the training set.

4.1 Feature (window size)

Performance was dependent on the window size (the number of left/right side words). Figure 3 depicts the performance obtained using various window sizes. The best performance was scored at the BOTH=6 setting. Therefore, this window size was used for the following experiments. These results also indicated that entire sentences (BOTH=∞) are unsuitable for this task.

4.2 Machine Learning Method

We compared various machine-learning methods from two points of view: accuracy and time. The result, presented in Table 2, shows that SVM with a polynomial kernel showed feasibility from both viewpoints of accuracy and the training time.

5 Experiments

We assessed the detection performance using actual influenza reports provided by the Japanese IDSC.

5.1 Comparable Methods

We compared the various methods as follows:

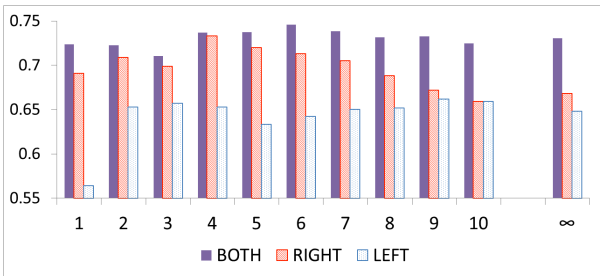


Figure 3: Window size and Accuracy (*F*-measure). RIGHT shows a method used only the right context. LEFT shows a method used only the left context. BOTH represents a method using both the right and left context. The number shows the window size. ∞ uses all words in each context direction.

Classifier	F-Measure	Training Time (sec)
AdaBoost (Freund 1996)	0.592	40.192
Bagging (Breiman 1996)	0.739	30.310
Decision Tree (Quinlan1993)	0.698	239.446
Logistic Regression	0.729	696.704
Naive Bayes	0. 741	**7.383**
Nearest Neighbor	0.695	22.441
Random Forest (Breiman 2001)	0.729	38.683
SVM (RBF kernel) (Cortes and Vapnik 1995)	0.738	92.723
SVM (polynomial kernel; d=2)	**0.756**	13.256

Table 2: Machine Learning Methods and Performance (*F*-measure and Training Time)

- **`TWEET-SVM`**: The proposed SVM-based method (window size = 6).
- **`TWEET-RAW`**: A simple frequency-based method. This approach outputs the relative frequency of word "*influenza*" appearing in Twitter.
- **`DRUG`**: The amounts of drug sales (sales of cold medicines). Statistics are provided by the Japanese Ministry of Health, Labor and Welfare.
- **`GOOGLE`**: Google flu trend detection (Japanese version). This method uses a query log of the Google search engine (Ginsberg et al., 2009)[5].

[5] http://www.google.org/flutrends/

5.2 Gold Standard and Test-Set

For gold standard data, we used data that are described in Section 2, as reported from IDSC. The report is released once a week. Therefore, the evaluation is done on a weekly basis.

We split the data into four seasons as follows:

- Season I: winter 2008,
- Season II: summer 2009,
- Season III: winter 2009,
- Season IV: summer 2010.

To investigate further detailed evaluations, we split the winters into two sub-seasons: **before the peak** and **after the peak**. We regard the peak point as the day with the highest number in that season. The statistics derived from the data are presented in Table 3.

Excessive News Period: In our experimental data, Season II and the earlier peak of Season III are special periods because news related to swine flu (H1N1 flu) is extremely hot in those seasons (Fig. 4). This paper calls them **Excessive News Periods**. We also investigated the results with and without the excessive news period.

Figure 4: A CNN news on "swine flu" in June 2009 (Season II in our experiment).

Experimental data include such excessive news periods.

5.3 Evaluation Metric

The evaluation metric is based on correlation (Pearson correlation) between the gold standard value and the estimated value.

5.4 Result

The results are presented in Table 4. In the non-excessive news period, the proposed method achieved the highest performance (0.890 correlation). This correlation is considerably higher than the query-based approach (**GOOGLE**), demonstrating the basic feasibility of the proposed approach. However, during the excessive news periods, the proposed method suffers from an avalanche of news, generating a news bias. This phenomenon is a remaining problem to be resolved in future studies.

6 Discussion

6.1 SVM-based Negative Filtering contributes to Performance

In most seasons, the proposed SVM approach (**TWEET-SVM**) shows higher correlation than the simple word lookup method (**TWEET-RAW**). The average improvement is 0.196 (max 0.56; min-0.009), which significantly boosts the correlation. This result demonstrates the basic feasibility of the proposed approach. In the future, more advantages attributable to the proposed approach can be obtained if the classification performance improves.

6.2 All Methods Suffer from News Bias in Excessive News Period

All methods expose the poor performance that prevails during the excessive news period (from Season II to Season III before the peak). Especially, tweet-based methods show dramatically reduced correlation, which indicates that Twitter is vulnerable to newswire bias.

One reason for that vulnerability is that Twitter is a kind of communication tool by which a tweet affects other people. Consequently, the possibility exists that a few tweets related to "flu" might spread widely, generating an explosive burst of influenza-related tweets. Future studies must address this burst phenomenon.

<table>
<tr><td colspan="5">All Season</td></tr>
<tr><td colspan="5">79 weeks (0.221)</td></tr>
<tr><td colspan="2">Season I

2008/11/9 - 2009/4/5</td><td>Season II

2009/4/12 - 2009/7/5</td><td colspan="2">Season III

2009/7/12 - 2010/2/14</td><td>Season IV

2010/2/21 - 2010/7/4</td></tr>
<tr><td colspan="2">22 weeks (0.423)</td><td rowspan="2">13 weeks
(0.553)</td><td colspan="2">26 weeks (0.388)</td><td rowspan="2">18 weeks
(0.468)</td></tr>
<tr><td>Before peak
2008/11/9-2009/1/25</td><td>After peak
2009/2/1-2009/4/5</td><td>Before peak
2009/7/12-2009/11/29</td><td>After peak
2009/12/6-2010/2/14</td></tr>
<tr><td>12weeks (0.576)</td><td>10 weeks
(0.632)</td><td></td><td>15 weeks
(0.514)</td><td>11 weeks
(0.602)</td><td></td></tr>
<tr><td colspan="2">Non-excessive news period</td><td colspan="2">Excessive news period</td><td colspan="2">Non-excessive news period</td></tr>
</table>

Table 3: Test-set Tracks and the number of data points (=weeks).
The number in the bracket indicates the statistical significance level.

		TWEET-RAW	TWEET-SVM (Proposed Method)	DRUG	GOOGLE
Excessive news period		0.001	0.060	**0.844**	**0.918**
Non- excessive news period		**0.831**	**0.890**	0.308	**0.847**
Season I		**0.683**	**0.816**	-0.208	**0.817**
	Before peak	**0.914**	**0.974**	-0.155	**0.962**
	After peak	**0.952**	**0.955**	0.557	**0.959**
Season II		-0.009	-0.018	0.406	0.232
Season III		0.382	**0.474**	**0.684**	**0.881**
	Before peak	0.390	0.474	**0.919**	**0.924**
	After peak	**0.960**	**0.944**	0.364	**0.936**
Season IV		0.391	**0.957**	0.130	**0.976**

Table 4: Results (Correlation Ratio).
The number in bold indicates the significance correlation (p=0.05). The number with underline indicates the highest value in each season.

6.3 Tweets have Advantages in Early Stage Detection

From practical viewpoints, the most important task is to detect influenza epidemics before the peak (early stage detection). Consequently, the correlation of the two seasons, Season I before the peak and Season III before the peak, presents the practical performance. Figure 5 portrays detailed results of all methods.

In Season I before the peak (Figure 5 Left), the proposed method (TWEET-SVM) shows the best performance among all methods.

In Season II before the peak (Figure 5 Right), all methods including the proposed method showed poor correlation because they are included in the excessive news periods. During that season, the newswires heavily reported the swine flu twice (April 2009 and May 2009). Because of this news, we can see two peaks in Twitter-based methods (TWEET-SVM and TWEET-RAW), which indicates that Twitter is more sensitive to the newswires.

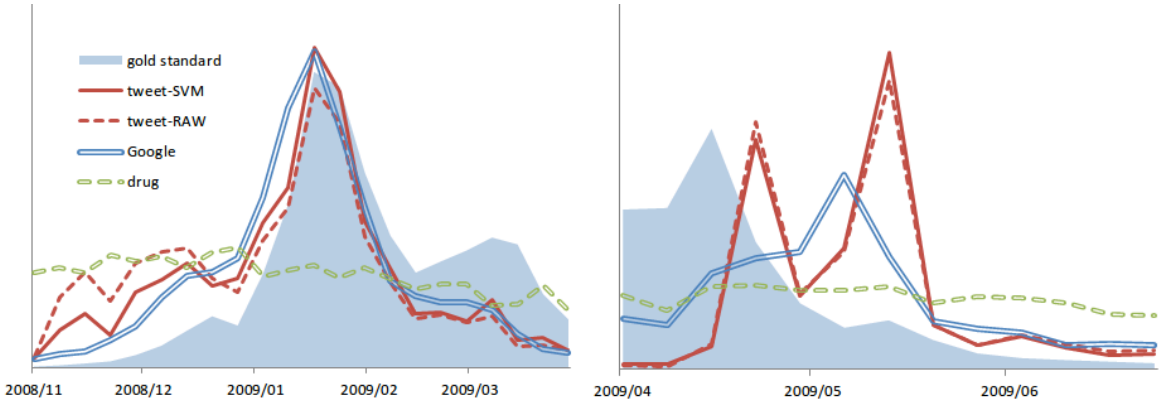


Figure 5: Predicted Values in Season I (Left) and Season II (Right):
the X-axis shows the date; the Y-axis shows the relative predicted value using each method.

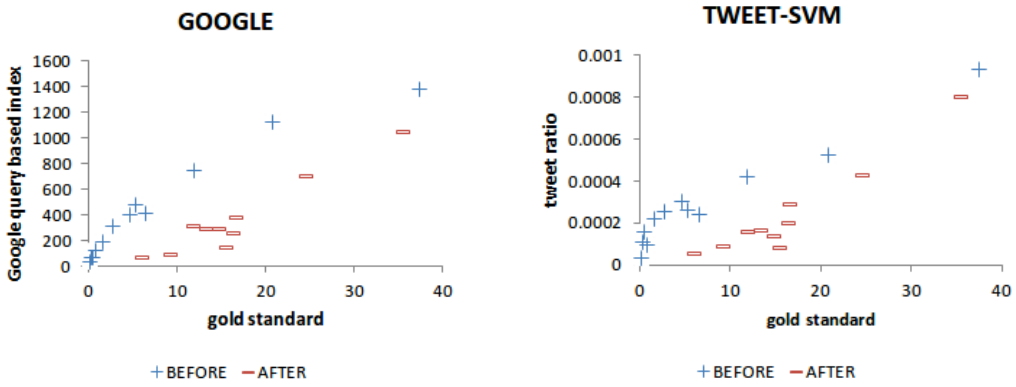


Figure 6: Patient Actions (Web Search Query and Tweet) is Sensitive before the Epidemic Peaks.
Distribution between the gold standard and Detected Values (Search Engine Query (Left) and Tweet (Right)): "+" denotes the distribution before the peak; "-" denotes the distribution after the peak.

6.4 Human Action is Sensitive before Epidemics

Figure 6 presents the distribution between the detected values (using **GOOGLE** and using **TWEET-SVM**) and the gold standard value (before the peak is shown by "+"; that after the peak is shown as "-"). Although the detected values fundamentally correlate with the gold standard, we can see different sensitivity before and after peak (The distribution before peak "+" is a higher value than after peak "-".).

Results show that human action, a web search (**GOOGLE**) and a tweet (**TWEET-SVM**), highly corresponds to the real influenza before the epidemic peaks, and vice versa. More acute detection is possible if we incorporate a model considering this aspect of human nature.

7 Related Works

The core technology of the proposed method is to classify whether the event is positive or negative. This task is similar to negation identification, which is a traditional topic, especially in medical fields. Therefore, we can find many previous studies of the topic in the relevant literature. An algorithm based approach, *NegEx* (Chapman et al., 2001), *Negfinder* (Mutalik et al., 2001), and *ConText* (Chapman et al., 2007), a machine learning based approach (Elkin et al., 2005; Huang and H.J. Lowe, 2007).

	Previous Negation (Syntactic)	This study: Negative Influenza (Semantic)
I caught a flu.	Positive sentence	Positive Influenza
I don't have the flu!	Negative sentence	Negative Influenza
I have enough flu drugs.	Positive sentence	Negative Influenza
I have not recovered from the flu.	Negative sentence	Positive Influenza

Table 5: Our target influenza negation (semantic) and previous negation (syntactic)

Although these approaches specifically examine the syntactic negation, this study detects the negative influenza, which is a specified semantic negation. Table 5 presents the difference between both negations. In general, the semantic operation is difficult in general. However, this paper revealed that the domain (influenza domain) specific semantic operation provides reasonable results.

Another aspect of this study is the target material, Twitter data, which have drawn much attention. Twitter can provide suitable material for many applications such as named entity recognition (NER) (Finin et al., 2010) and sentiment analysis (Barbosa and Feng, 2010). Although these studies specifically examine the fundamental NLP techniques, this study directly targets an NLP application that can contribute to our daily life.

8 Conclusion

This paper proposed a new Twitter-based influenza epidemics detection method, which relies on the Natural Language Processing (NLP). Our proposed method could successfully filter out the negative influenza tweets (*f*-measure=0.76), which are posted by the ones who did not actually catch the influenza. The experiments with the test data empirically demonstrate that the proposed method detects influenza epidemics with high correlation (correlation ratio=0.89), which outperforms the state-of-the-art Google method. This result shows that Twitter texts precisely reflect the real world, and that the NLP technique can extract the useful information from Twitter streams.

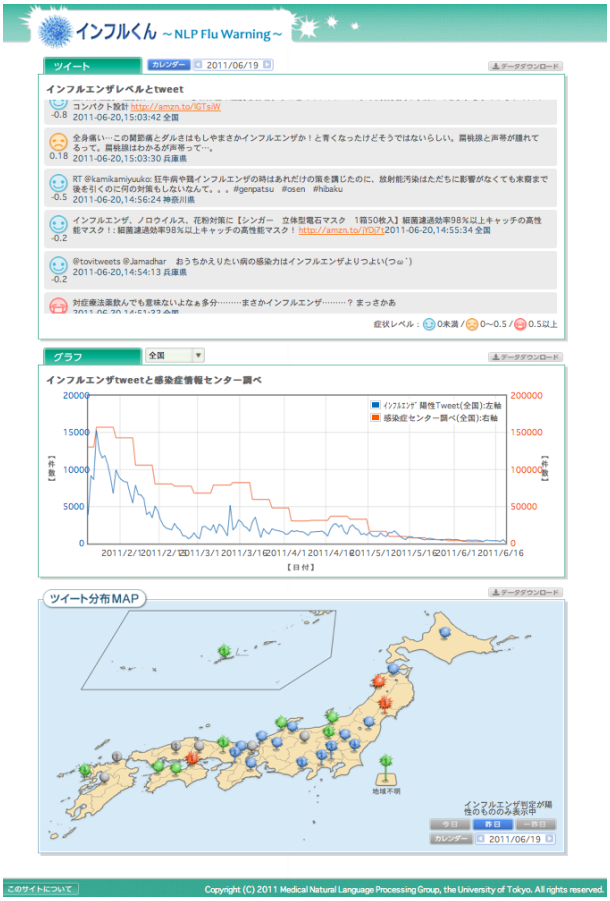


Figure 7: An influenza severance system "*INFLU kun*" using the proposed method is available at http://mednlp.jp/influ/.

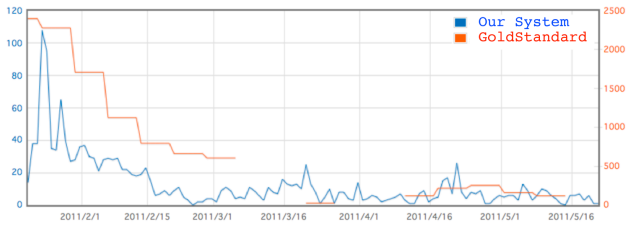


Figure 8: The Timeline of Influenza Epidemics in Fukushima. While the Infection Disease Surveillance Center (IDSC) sometimes stops (gold standard) due to the Great East Japan Earthquake, the proposed system could continue to work (Our System).

Available Resources

Corpus: The corpus of this study is provided at the http://mednlp.jp/~aramaki/KAZEMIRU/.

Web System: The web service is also released at http://mednlp.jp/influ/ (Figure 7 and Figure 8).

References

Barbosa, L. and J. Feng. 2010. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In Proc. 23rd Intl. Conf. on Computational Linguistics (COLING).

Boyd, D., S. Golder, and G. Lotan. 2010. Tweet, tweet, retweet: Conversational aspects of retweeting on Twitter. In Proc. HICSS43.

Breiman L. Random Forests. 2001. Machine learning, 45(1): 5–32.

Breiman, L. Bagging predictors. 1996. Machine learning, 24(2):123–140.

Cortes C. and V. Vapnik. 1995. Support vector networks. In Machine Learning, pp. 273–297.

Chapman, W., W. Bridewell, P. Hanbury, G.F. Cooper, and B. Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. Journal of Biomedical Informatics, 5:301-310.

Chapman, W., J. Dowling, and D. Chu. 2007. ConText: An algorithm for identifying contextual features from clinical text. Biological, translational, and clinical language processing (BioNLP2007), pp. 81–88.

Elkin, P.L., S.H. Brown, B.A. Bauer, C.S. Husser, W. Carruth, L.R. Bergstrom, and D.L. Wahner-Roedler. 2005. A controlled trial of automated classification of negation from clinical notes. BMC Medical Informatics and Decision Making 5:13.

Espino, J., W. Hogan, and M. Wagner. 2003. Telephone triage: A timely data source for surveillance of influenza-like diseases. In Proc. of Annual Symposium of AMIA, pp. 215–219.

Finin, T., W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze. 2010. Annotating named entities in Twitter data with crowdsourcing. In Proc. NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk (CSLDAMT '10), pp. 80-88.

Freund, Y. and R. Schapire. 1996. Experiments with a new boosting algorithm. In Machine Learning Intl. Workshop, pp.148–156.

Ginsberg, J., M.H. Mohebbi, R.S. Patel, and L. Brammer. 2009. Detecting influenza epidemics using search engine query data, Nature Vol. 457 (19).

Huang, Y. and H.J. Lowe. 2007. A novel hybrid approach to automated negation detection in clinical radiology reports. Journal of the American Medical Informatics Association, 14(3):304-311.

Huberman, B. and D. R. F. Wu. 2009. Social networks that matter: Twitter under the microscope. First Monday, Vol. 14.

Hulth, A., G. Rydevik, and A. Linde. 2009. Web Queries as a Source for Syndromic Surveillance. PLoS ONE 4(2).

Johnson, HA., MM. Wagner, WR. Hogan, W. Chapman, RT. Olszewski, J. Dowling, and G. Barnas. 2004. Analysis of Web access logs for surveillance of influenza. Stud. Health Technol. Inform. 107(Pt 2):1202-1206.

Magruder, S. 2003. Evaluation of over-the-counter pharmaceutical sales as a possible early warning indicator of human disease. Johns Hopkins University APL Technical Digest 24:349–353.

Milstein, S., A. Chowdhury, G. Hochmuth, B. Lorica, and R. Magoulas. 2008. Twitter and the micro-messaging revolution: Communication, connections, and immediacy, 140 characters at a time. O'Reilly Media.

Mutalik, P.G., A. Deshpande, and P.M. Nadkarni. 2001. Use of general purpose negation detection to augment concept indexing of medical documents: A quantitative study using theUMLS. Journal of the American Medical Informatics Association, 8(6):598-609.

Paul, MJ. and M. Dredze. 2011. You Are What You Tweet: Analyzing Twitter for Public Health. In Proc. of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM).

Polgreen, PM., Y. Chen, D.M. Pennock, and F.D. Nelson. 2008. Using Internet Searches for Influenza Surveillance, Clinical Infectious Diseases Vol. 47 (11) pp. 1443-1448.

Quinlan. J. 1993. C4. 5: programs for machine learning. Morgan Kaufmann.

Sakaki, T., M. Okazaki, and Y. Matsuo. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors, in Proc. of Conf. on World Wide Web (WWW).

A Simple Word Trigger Method for Social Tag Suggestion

Zhiyuan Liu, Xinxiong Chen and **Maosong Sun**
Department of Computer Science and Technology
State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
Tsinghua University, Beijing 100084, China
{lzy.thu, cxx.thu}@gmail.com, sms@tsinghua.edu.cn

Abstract

It is popular for users in Web 2.0 era to freely annotate online resources with tags. To ease the annotation process, it has been great interest in automatic tag suggestion. We propose a method to suggest tags according to the text description of a resource. By considering both the description and tags of a given resource as summaries to the resource written in two languages, we adopt word alignment models in statistical machine translation to bridge their *vocabulary gap*. Based on the translation probabilities between the words in descriptions and the tags estimated on a large set of description-tags pairs, we build a word trigger method (WTM) to suggest tags according to the words in a resource description. Experiments on real world datasets show that WTM is effective and robust compared with other methods. Moreover, WTM is relatively simple and efficient, which is practical for Web applications.

1 Introduction

In Web 2.0, Web users often use tags to collect and share online resources such as Web pages, photos, videos, movies and books. Table 1 shows a book entry annotated with multiple tags by users[1]. On the top of Table 1 we list the title and a short introduction of the novel "The Count of Monte Cristo". The bottom half of Table 1 shows the annotated tags, each of which is followed by a number in bracket, the total number of users who use the tag to annotate this book. Since the tags of a resource are annotated collaboratively by multiple users, we also name these tags as *social* tags. For a resource, we refer to the additional information, such as the title and introduction of a book, as *description*, and the user-annotated social tags as *annotation*.

Description
`Title:` The Count of Monte Cristo
`Intro:` *The Count of Monte Cristo* is one of the most popular fictions by Alexandre Dumas. The writing of the work was completed in 1844. ...
Annotation
Dumas (2748), Count of Monte Cristo (2716), foreign literature (1813), novel (1345), France (1096), classic (1062), revenge (913), famous book (759), ...

Table 1: An example of social tagging. The number in the bracket after each tag is the total count of users that annotate the tag on this book.

Social tags concisely indicate the main content of the given resource, and potentially reflect user interests. Social tagging has thus been widely studied and successfully applied in recommender systems (Eck et al., 2007; Yanbe et al., 2007; Zhou et al., 2010), trend detection and tracking (Hotho et al., 2006), personalization (Wetzker et al., 2010), advertising (Mirizzi et al., 2010), etc.

The task of automatic social tag suggestion is to automatically recommend tags for a user when he/she wants to annotate a resource. Social tag suggestion, as a crucial component for social tagging systems, can help users annotate resources. Moreover, social tag suggestion is usually considered as an equivalent problem to modeling social

[1]The original record is obtained from the book review website Douban (`www.douban.com`) in Chinese. Here we translate it to English for comprehension.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1577–1588,
Edinburgh, Scotland, UK, July 27–31, 2011.

tagging behaviors, which is playing a more and more important role in social computing and information retrieval (Wang et al., 2007).

Most online resources contain descriptions, which usually contain much resource information. For example, on a book review website, each book entry contains a title, the author(s) and an introduction of the book. Some researchers thus propose to automatically suggest tags based on resource descriptions, which are collectively known as the *content-based approach*.

One may think to suggest tags by selecting important words from descriptions. This is far from enough because descriptions and annotations are using diverse vocabularies, usually referred to as a *vocabulary gap* problem. Take the book entry in Table 1 for instance, the word "popular" used in the description contrasts the tags "classic" and "famous book" in the annotation; the word "novel" is used in the description, while most users annotate with the tag "fiction". The vocabulary gap usually reflects in two main issues:

- Some tags in the annotation do appear in the corresponding description, but they may not be statistically significant.

- Some tags may even not appear in the description.

It is not trivial to reduce the vocabulary gap and find the semantic correspondence between descriptions and annotations. By regarding both the description and the annotation as *parallel* summaries of a resource, we use word alignment models in statistical machine translation (SMT) (Brown et al., 1993) to estimate the translation probabilities between the words in descriptions and annotations. SMT has been successfully applied in many applications to bridge vocabulary gap. For detailed descriptions of related work, readers can refer to Section 2.2. In this paper, besides employing word alignment models to social tagging, we also propose a method to efficiently build description-annotation pairs for sufficient learning translation probabilities by word alignment models.

Based on the learned translation probabilities between words in descriptions and annotations, we regard the tagging behavior as a word trigger process:

1. A user reads the resource description to realize its substance by seeing some important words in the description.

2. Triggered by these important words, the user translates them into the corresponding tags, and annotates the resource with these tags.

Based on this perspective, we build a simple word trigger method (WTM) for social tag suggestion. In Fig. 1, we use a simple example to show the basic idea of using word trigger for social tag suggestion. In this figure, some words in the first sentence of the book description in Table 1 are triggered to the tags in annotation.

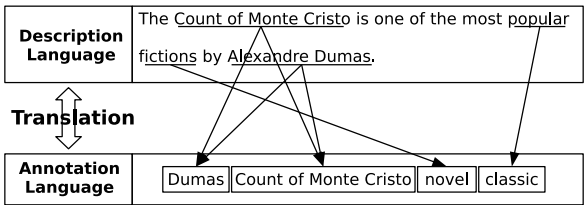


Figure 1: An example of the word trigger method for suggesting tags given a description.

2 Related Work

2.1 Social Tag Suggestion

Previous work has been proposed to automatic social tag suggestion.

Many researchers built tag suggestion systems based on *collaborative filtering* (CF) (Herlocker et al., 1999; Herlocker et al., 2004), a widely used technique in recommender systems (Resnick and Varian, 1997). These collaboration-based methods typically base their suggestions on the tagging history of the given resource and user, without considering resource descriptions. FolkRank (Jaschke et al., 2008) and Matrix Factorization (Rendle et al., 2009) are representative CF methods for social tag suggestion. Most of these methods suffer from the *cold-start problem*, i.e., they are not able to perform effective suggestions for resources that no one has annotated yet.

The content-based approach for social tag suggestion remedies the cold-start problem of the

collaboration-based approach by suggesting tags according to resource descriptions. Therefore, the content-based approach plays an important role in social tag suggestion.

Some researchers regarded social tag suggestion as a classification problem by considering each tag as a category label (Ohkura et al., 2006; Mishne, 2006; Lee and Chun, 2007; Katakis et al., 2008; Fujimura et al., 2008; Heymann et al., 2008). Various classifiers such as Naive Bayes, kNN, SVM and neural networks have been explored to solve the social tag suggestion problem.

There are two issues emerging from the classification-based methods:

- The annotations provided by users are noisy, and the classification-based methods can not handle the issue well.
- The training cost and classification cost of many classification-based methods are usually in proportion to the number of classification labels. These methods may thus be inefficient for a real-world social tagging system, where hundreds of thousands of unique tags should be considered as classification labels.

Inspired by the popularity of latent topic models such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003), various methods have been proposed to model tags using generative latent topic models. One intuitive approach is assuming that both tags and words are generated from the same set of latent topics. By representing both tags and descriptions as the distributions of latent topics, this approach suggests tags according to their likelihood given the description (Krestel et al., 2009; Si and Sun, 2009). Bundschus et al. (2009) proposed a joint latent topic model of users, words and tags. Iwata et al. (2009) proposed an LDA-based topic model, Content Relevance Model (CRM), which aimed at finding the content-related tags for suggestion. Empirical experiments showed that CRM outperformed both classification methods and Corr-LDA (Blei and Jordan, 2003), a generative topic model for contents and annotations.

Most latent topic models have to pre-specify the number of topics before training. We can either use cross validation to determine the optimal number of topics or employ the infinite topic models, such as Hierarchical Dirichlet Process (HDP) (Teh et al., 2006) and nested Chinese Restaurant Process (Blei et al., 2010), to automatically adjust the number of topics during training. Both solutions are usually computationally complicated. What is more important, topic-based methods suggest tags by measuring the topical relevance of tags and resource descriptions. The latent topics are of concept-level which are usually too general to precisely suggest those specific tags such as named entities, e.g., the tags "Dumas" and "Count of Monte Cristo" in Table 1. To remedy the problem, Si et al. (2010) proposed a generative model, Tag Allocation Model (TAM), which considers the words in descriptions as the possible topics to generate tags. However, TAM assumes each tag can only have at most one word as its reason. This is against the fact that a tag may be annotated triggered by multiple words in the description.

It should also be noted that social tag suggestion is different from automatic keyphrase extraction (Turney, 2000; Frank et al., 1999; Liu et al., 2009a; Liu et al., 2010b; Liu et al., 2011). Keyphrase extraction aims at selecting terms from the given document to represent the main topics of the document. On the contrary, in social tag suggestion, the suggested tags do not necessarily appear in the given resource description. We can thus regard social tag suggestion as a task of selecting appropriate tags from a controlled tag vocabulary for the given resource description.

2.2 Applications of SMT

SMT techniques have been successfully used in many tasks of information retrieval and natural language processing to bridge the vocabulary gap between two types of objects. Some typical tasks are document information retrieval (Berger and Lafferty, 1999; Murdock and Croft, 2004; Karimzadehgan and Zhai, 2010), question answering (Berger et al., 2000; Echihabi and Marcu, 2003; Soricut and Brill, 2006; Riezler et al., 2007; Surdeanu et al., 2008; Xue et al., 2008), query expansions (Riezler et al., 2007; Riezler et al., 2008; Riezler and Liu, 2010), paraphrasing (Quirk et al., 2004; Zhao et al., 2010a; Zhao et al., 2010b), summarization (Banko et al., 2000), collocation extraction (Liu et al., 2009b;

Liu et al., 2010c), keyphrase extraction (Liu et al., 2011), sentiment analysis (Dalvi et al., 2009), computational advertising (Ravi et al., 2010), and image/video annotation and retrieval (Duygulu et al., 2002; Jeon et al., 2003).

3 Word Trigger Method for Social Tag Suggestion

3.1 Method Framework

We describe the word trigger method (WTM) for social tag suggestion as a 3-stage process:

1. Preparing description-annotation pairs. Given a collection of annotated resources, we first prepare description-annotation pairs for learning translation probabilities using word alignment models.

2. Learning a translation model. Given a collection of description-annotation pairs, we adopt IBM Model-1, a widely used word alignment model, to learn the translation probabilities between words in descriptions and tags in annotations.

3. Suggesting tags given a resource description. After building translation probabilities between words and tags, given a resource description, we first compute the trigger power of each word in the description and then suggest tags according to their translation probabilities from the triggered words.

Before introducing the method in details, we introduce the notations. In a social tagging system, a resource is denoted as $r \in R$, where R is the set of all resources. Each resource contains a description and an annotation containing a set of tags. The description d_r of resource r can be regarded as a bag of words $\mathbf{w}_r = \{(w_i, e_i)\}_{i=1}^{N_r}$, where e_i is the count of word w_i and N_r is the number of unique words in r. The annotation a_r of resource r is represented as $\mathbf{t}_r = \{(t_i, e_i)\}_{i=1}^{M_r}$, where e_i is the count of tag t_i and M_r is the number of unique tags for r.

3.2 Preparing Description-Annotation Pairs

Learning translation probabilities requires a parallel training dataset consisting of a number of aligned sentence pairs. We assume the description and the annotation of a resource as being written in two distinct languages. We thus prepare our parallel training dataset by pairing descriptions with annotations.

The annotation of a resource is a bag of tags with no position information. We thus select IBM Model-1 (Brown et al., 1993) for training, which does not take word position information into account on both sides for each aligned pair.

In a social tagging system, the length of a resource description is usually limited to hundreds of words. Meanwhile, it is common that some popular resources are annotated by multiple users with thousands of tags. For example, the tag *Dumas* is annotated by $2,748$ users for the book in Table 1. We have to deal with the length-unbalance between a resource description and its corresponding annotation for two reasons.

- It is impossible to list all annotated tags on the annotation side of a description-annotation pair. The performance of word alignment models will also suffer from the unbalanced length of sentence pairs in the parallel training data set (Och and Ney, 2003).

- Moreover, the annotated tags may have different importance for the resource. It would be unfair to treat these tags without distinction.

Here we propose a sampling method to prepare length-balanced description-annotation pairs for word alignment. The basic idea is to sample a bag of tags from the annotation according to tag weights and make the generated bag of tags with comparable length with the description.

We consider two parameters when sampling tags. First, we have to select a **tag weighting type** for sampling. In this paper, we investigate two straightforward sampling types, including tag frequency (TF_t) within the annotation and tag-frequency inverse-resource-frequency (TF-IRF_t). Given resource r, TF_t and TF-IRF_t of tag t are defined as $\text{TF}_t = e_t / \sum_t e_t$ and $\text{TF-IRF}_t = e_t / \sum_t e_t \times \log\left(|R| / |\sum_{r \in R} I_{e_t > 0}|\right)$, where $|\sum_{r \in R} I_{e_t > 0}|$ indicates the number of resources that have been annotated with tag t.

Another parameter is the **length ratio** between the description and the sampled annotation. We denote the ratio as $\delta = |\mathbf{w}_r| / |\mathbf{t}_r|$, where $|\mathbf{w}_r|$ is the number of words in the description and $|\mathbf{t}_r|$ is the number of tags in the annotation.

3.3 Learning Translation Probabilities Using Word Alignment Models

Suppose the source language is resource description and the target language is resource annotation. In IBM Model-1, the relationship of the source language $\mathbf{w} = w_1^J$ and the target language $\mathbf{t} = t_1^I$ is connected via a hidden variable describing an alignment mapping from source position j to target position a_j:

$$\Pr(w_1^J|t_1^I) = \sum_{a_1^J} \Pr(w_1^J, a_1^J|t_1^I). \quad (1)$$

The alignment a_1^J also contains empty-word alignments $a_j = 0$ which align source words to the an empty word. IBM Model-1 can be trained using Expectation-Maximization (EM) algorithm in an unsupervised fashion, and obtains the translation probabilities of two vocabularies, i.e., $\Pr(w|t)$, where t is a tag and w is a word.

IBM Model-1 only produces one-to-many alignments from source language to target language. The learned model is thus asymmetric. We will learn translation models on two directions: one is regarding descriptions as the source language and annotations as the target language, and the other is in reverse direction of the pairs. We denote the first model as $\Pr_{d2a}$ and the latter as $\Pr_{a2d}$. We further define $\Pr(t|w)$ as the harmonic mean of the two models:

$$\Pr(t|w) \propto \Big(\lambda/\Pr{}_{d2a}(t|w)+(1-\lambda)/\Pr{}_{a2d}(t|w)\Big)^{-1}, \quad (2)$$

where λ is the harmonic factor to combine the two models. When $\lambda = 1$ or $\lambda = 0$, it simply uses model $\Pr_{d2a}$ or $\Pr_{a2d}$ correspondingly.

3.4 Tag Suggestion Using Triggered Words and Translation Probabilities

When given the description of a resource, we can rank tags by computing the scores:

$$\Pr(t|d = \mathbf{w}_d) = \sum_{w \in \mathbf{w}_d} \Pr(t|w) \Pr(w|d), \quad (3)$$

in which $\Pr(w|d)$ is the trigger power of the word w in the description, which indicates the importance of the word. According to the ranking scores, we can suggest the top-ranked tags to users.

Here we explore three methods to compute the trigger power of a word in a resource description: TF-IRF$_w$, TextRank and their product. TF-IRF$_w$ and TextRank are two most widely adopted methods for keyword extraction.

Similar to TF-IRF$_t$ mentioned in Section 3.2, TF-IRF$_w$ considers both the local importance (TF$_w$) and global specification (IRF$_w$).

TextRank (Mihalcea and Tarau, 2004) is a graph-based method to compute term importance. Given a resource description, TextRank first builds a term graph by connecting the terms in the description according to their semantic relations, and then run PageRank algorithm (Page et al., 1998) to measure the importance of each term in the graph. Readers can refer to (Mihalcea and Tarau, 2004) for detailed information.

We also use the product of TF-IRF$_w$ and TextRank to weight terms, which potentially takes both global information and term relations into account.

Emphasize Tags Appearing In Description for WTM (EWTM) In some social tagging systems, the tags that appear in the resource description are more likely to be selected by users for annotation. Therefore, we propose to emphasize the tags in the description by ranking tags as follows

$$\Pr(t|d) = \sum_{w \in \mathbf{w}_d} \big(\gamma I_t(w) + (1-\gamma) \Pr(t|w)\big) \Pr(w|d), \quad (4)$$

where $I_t(w)$ is an indicator function which gets value 1 when $t = w$ and 0 when $t \neq w$; and γ is the smooth factor with range $\gamma \in [0.0, 1.0]$. When $\gamma = 1.0$, it suggests tags simply according to their trigger powers within the description, while when $\gamma = 0.0$, it does not emphasize the tags appearing in the description and just suggests according to their translation probabilities. In Section 4.4, we will show the performance of EWTM.

4 Experiments

4.1 Datasets and Evaluation Metrics

Datasets In our experiments, we select two real world datasets which are of diverse properties to evaluate our methods. In Table 2 we show the detailed statistical information of the two datasets.

Data	R	W	T	$\bar{N}_w$	$\bar{N}_t$
BOOK	70,000	174,748	46,150	211.6	3.5
BIBTEX	158,924	91,277	50,847	5.8	2.7

Table 2: Statistical information of two datasets. R, W, T, $\bar{N}_w$ and $\bar{N}_t$ are the number of resources, the vocabulary of descriptions, the vocabulary of tags, the average number of words in each description and the average number of tags in each resource, respectively.

The first dataset, denoted as BOOK, is obtained from a popular Chinese book review website `www.douban.com`, which contains the descriptions of books and the tags collaboratively annotated by users. The second dataset, denoted as BIBTEX, is obtained from an English online bibliography website `www.bibsonomy.org`[2]. The dataset contains the descriptions for academic papers (including the title and note for each paper) and the tags annotated by users. As shown in Table 2, the average length of descriptions in the BIBTEX dataset is much shorter than the BOOK dataset. Moreover, the BIBTEX dataset does not provide how many times each tag is annotated to a resource.

Evaluation Metrics We use precision, recall and F-measure to evaluate the performance of tag suggestion methods. For a resource, we denote the original tags (gold standard) as T_a, the suggested tags as T_s, and the correctly suggested tags as $T_s \cap T_a$. Precision, recall and F-measure are defined as

$$p = \frac{|T_s \cap T_a|}{|T_s|}, \quad r = \frac{|T_s \cap T_a|}{|T_a|}, \quad F = \frac{2pr}{(p+r)}. \tag{5}$$

The final evaluation scores are computed by micro-averaging (i.e., averaging on resources of test set). We perform 5-fold cross validation for each method on all two datasets. In experiments, the number of suggested tags M ranges from 1 to 10.

4.2 Comparing Results

Baseline Methods We select four content-based algorithms as the baselines for comparison: Naive Bayes (NB) (Manning et al., 2008), k nearest neighbor algorithm (kNN) (Manning et al., 2008), Content Relevance (CRM) model (Iwata et al., 2009) and Tag Allocation Model (TAM) (Si et al., 2010).

NB and kNN are two representative classification methods. NB is a simple generative model, which models the probability of each tag t given description d as

$$\Pr(t|d) \propto \Pr(t) \prod_{w \in d} \Pr(w|t). \tag{6}$$

$\Pr(t)$ is estimated by the frequency of the resources annotated with the tag t. $\Pr(w|t)$ is estimated by the frequency of the word w in the resource descriptions annotated with the tag t. kNN is a widely used classification method for tag suggestion, which recommends tags to a resource according to the annotated tags of similar resources measured using vector space models (Manning et al., 2008).

CRM and TAM are selected to represent topic-based methods for tag suggestion. CRM is an LDA-based generative model. The number of latent topics K is the key parameter for CRM. In experiments, we evaluated the performance of CRM with different K values, and here we only show the best one obtained by setting $K = 1,024$. TAM is also a generative model which considers the words in descriptions as the topics to further generate tags for the resource. We set parameters for TAM as in (Si et al., 2010). For comparison, we denote our method as WTM.

Complexity Analysis We compare the complexity of these methods. We denote the number of training iterations in CRM, TAM and WTM as I [3], and the number of topics in CRM as K. For the training phase, the complexity of NB is $O(R\bar{N}_w\bar{N}_t)$, kNN is $O(1)$, TAM is $O(IR\bar{N}_w\bar{N}_t)$, CRM is $O(IKR\bar{N}_w\bar{N}_t)$, and WTM is $O(IR\bar{N}_w\bar{N}_t)$[4]. When suggesting for a given resource description with length N_w, the complexity of NB is $O(N_wT)$, kNN is $O(R\bar{N}_w\bar{N}_t)$, CRM is $O(IKN_wT)$, TAM

[2]The dataset can be obtained from `http://www.kde.cs.uni-kassel.de/bibsonomy/dumps`

[3]In fact, the numbers of iterations of the three methods are different from each other. For simplicity, here we denote them using the same notation.

[4]In more detail, the training phase of WTM contains preparing parallel training dataset with $O(R\bar{N}_t)$ and learning translation probabilities using word alignment models with $O(IR\bar{N}_w\bar{N}_t)$, where I is the number of iterations for learning translation probabilities, and $\bar{N}_t$ is the average number of tags for each resource after sampling.

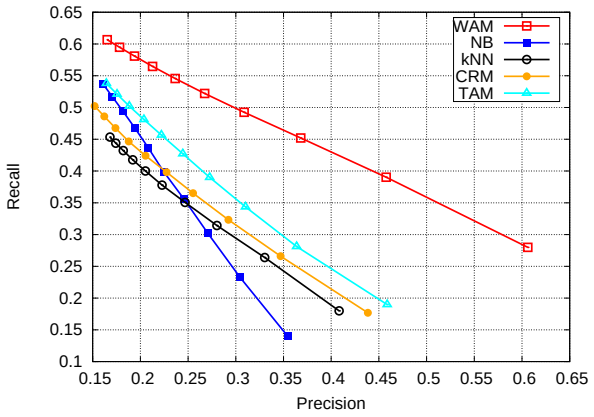
is $O(IN_wT)$ and WTM is $O(N_wT)$. From the analysis, we can see that WTM is a relatively simple method for both training and suggestion. This is especially valuable because WTM also shows good effectiveness for tag suggestion compared with other methods as we will shown later.

Parameter Settings We use GIZA++ (Och and Ney, 2003)[5] as IBM Model-1 to learn translation probabilities using description-annotation pairs for WTM. The experimental results of WTM are obtained by setting parameters as follows: tag weighting type as TF-IRF$_t$, length ratio $\delta = 1$, harmonic factor $\lambda = 0.5$ and the type of word trigger strength as TF-IRF$_w$. The influence of parameters to WTM can be found in Section 4.3.

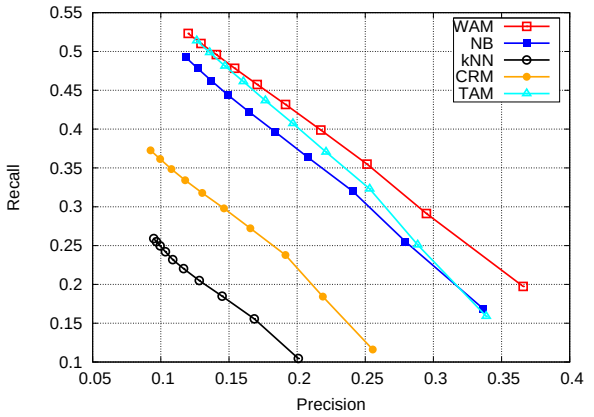
Experiment Results and Analysis In Fig. 2 we show the precision-recall curves of NB, kNN, CRM and WTM on two datasets. Each point of a precision-recall curve represents different numbers of suggested tags from $M = 1$ (bottom right, with higher precision and lower recall) to $M = 10$ (upper left, with higher recall but lower precision) respectively. The closer the curve to the upper right, the better the overall performance of the method. From Fig. 2, we observe that:

- WTM consistently performs the best on both datasets. This indicates that WTM is robust and effective for tag suggestion.
- The advantage of WTM is more significant on the BOOK dataset. The reason is that WTM can take a good advantage of annotation count information of tags compared to other methods.
- The average length of resource descriptions is short in the BIBTEX dataset, which makes it difficult to determine the trigger powers of words. But even on the BIBTEX dataset with no count information of tags, WTM still outperforms other methods especially when recommending first several tags.

To further demonstrate the performance of WTM and other baseline methods, in Table 3 we show the



(a) BOOK



(b) BIBTEX

Figure 2: Performance comparison between NB, kNN, CRM, TAM and WTM on two datasets.

precision, recall and F-measure of NB, kNN, CRM, TAM and WTM on BOOK dataset when suggesting $M = 3$ tags[6]. Due to the limit of space, we only show the variance of F-measure. In fact, WTM achieves its best performance when $M = 2$, where the F-measure of WTM is 0.370, outperforming both CRM ($F = 0.263$) and TAM ($F = 0.277$) by about 10%.

An Example In Table 4 we show top 10 tags suggested by NB, CRM, TAM and WTM for the book in Table 1. The number in bracket after the name of each method is the count of correctly suggested tags. The correctly suggested tags are marked in bold face. We select not to show

[5]GIZA++ is freely available on `code.google.com/p/giza-pp`. The toolkit is widely used for word alignment in SMT. In this paper, we use the default setting of parameters for training.

[6]We select to show this number because it is near the average number of tags for BOOK dataset

Method	Precision	Recall	F-measure
NB	0.271	0.302	0.247 ± 0.004
kNN	0.280	0.314	0.258 ± 0.002
CRM	0.292	0.323	0.266 ± 0.004
TAM	0.310	0.344	0.283 ± 0.001
WTM	**0.368**	**0.452**	$\mathbf{0.355 \pm 0.002}$

Table 3: Comparing results of NB, kNN, CRM, TAM and WTM on BOOK dataset when suggesting $M = 3$ tags.

the results of kNN because the tags suggested by kNN are totally unrelated to the book due to the insufficient finding of nearest neighbors.

From Table 4, we observe that NB, CRM and TAM, as generative models, tend to suggest general tags such as "novel", "literature", "classic" and "France", and fail in suggesting specific tags such as "Alexandre Dumas" and "Count of Monte Cristo". On the contrary, WTM succeeds in suggesting both general and specific tags related to the book.

NB (+6): **novel**, **foreign literature**, **literature**, **history**, Japan, **classic**, **France**, philosophy, America, biography
CRM (+5): **novel**, **foreign literature**, **literature**, biography, philosophy, culture, **France**, British, comic, **history**
TAM (+5): **novel**, sociology, finance, **foreign literature**, **France**, **literature**, biography, **France literature**, comic, China
WTM (+7): **novel**, **Alexandre Dumas**, **history**, **Count of Monte Cristo**, **foreign literature**, biography, **suspense**, comic, America, **France**

Table 4: Top 10 tags suggested by NB, CRM, TAM and WTM for the book in Table 1.

In Table 5, we list four important words (using TF-IRF$_w$ as weighting metric) of the description and their corresponding tags with the highest translation probabilities. The values in brackets are the probability of tag t given word w, $\Pr(t|w)$. For each word, we eliminated the tags with the probability less than 0.1. We can see that the translation probabilities can map the words in descriptions to their semantically corresponding tags in annotations.

Count of Monte Cristo: Count of Monte Cristo (0.728), Alexandre Dumas (0.270), . . .
Alexandre Dumas: Alexandre Dumas (0.966), . . .
revenge: foreign literature (0.168), classic (0.130), martial arts (0.123), Alexandre Dumas (0.122), . . .
France: France (0.99), . . .

Table 5: Four important words (in bold face) in the book description in Table 1 and their corresponding tags with the highest translation probabilities.

4.3 Parameter Influences

We explore the parameter influences to WTM for social tag suggestion. The parameters include harmonic factor, length ratio, tag weighting types, and types of word trigger strength. When investigating one parameter, we set other parameters to be the values inducing the best performance as mentioned in Section 4.2. Finally, we also investigate the influence of training data size for suggestion performance. In experiments we find that WTM reveals similar trends on both the BOOK dataset and the BIBTEX dataset. We thus only show the experimental results on the BOOK dataset for analysis.

Harmonic Factor In Fig. 3 we investigate the influence of harmonic factor via the curves of F-measure of WTM versus the number of suggested tags on the BOOK dataset when harmonic factor λ ranges from 0.0 to 1.0. As shown in Section 3.3, harmonic factor λ controls the proportion between model $\Pr_{d2a}$ and $\Pr_{a2d}$.

From Fig. 3, we observe that neither single model $\Pr_{d2a}$ ($\lambda = 1.0$) nor $\Pr_{a2d}$ ($\lambda = 0.0$) achieves the best performance. When the two models are combined by harmonic mean, the performance is consistently better, especially when λ ranges from 0.2 to 0.6. This is reasonable because IBM Model-1 constrains that only the term in source language can be aligned to multiple terms in target language, which makes the translation probability learned by a single model be asymmetric.

Length Ratio Fig. 4 shows the influence of length ratios on the BOOK dataset. From the figure, we observe that the performance for tag suggestion is robust as the length ratio varies, except when the ratio breaks the default restriction of GIZA++ (i.e.,

$\delta = 10$)[7].

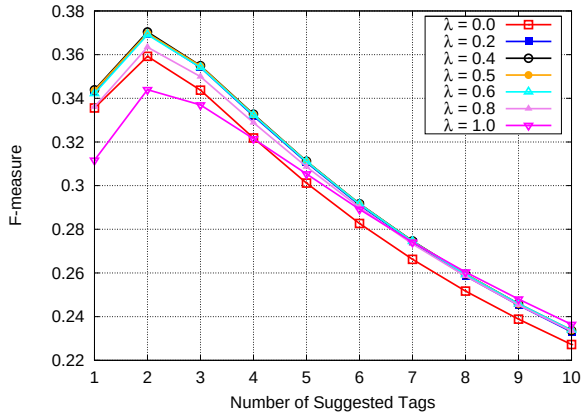


Figure 3: F-measure of WTM versus the number of suggested tags on the BOOK dataset when harmonic factor λ ranges from 0.0 to 1.0.

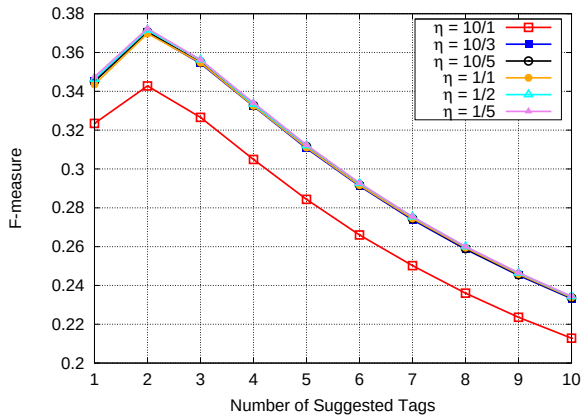


Figure 4: F-measure of WTM versus the number of suggested tags on the BOOK dataset when length ratio δ ranges from $10/1$ to $1/5$.

Tag Weighting Types The influence of two weighting types, TF_t and TF-IRF_t, on social tag suggestion when $M = 3$ on the BOOK dataset is shown in Table 6. TF-IRF_t tends to select the tags more specific to the resource while TF_t tends to select the most popular tags, because the latter does not consider global information (the IRF_t part).

[7]GIZA++ restricts the values of length ratio within $[\frac{1}{9}, 9]$ by setting parameter `maxfertility=10`. From Fig. 4, we can see when $\delta = 10$, the performance becomes much worse since GIZA++ will cut off the sentences out of range.

Table 6 verifies the analysis, where TF-IRF_t is slightly better than TF_t.

Weighting	Precision	Recall	F-measure
TF_t	0.356	0.437	0.342 ± 0.002
TF-IRF_t	0.368	0.452	0.355 ± 0.002

Table 6: Evaluation results for different tag weighting types when $M = 3$ on the BOOK dataset.

Methods for Computing Word Trigger Power In Table 7, we show the performance of social tag suggestions on the BOOK dataset with different methods for computing word trigger power. From the table, we can see that there is not significant difference between TF-IRF_w and the product of TF-IRF_w and TextRank, while TextRank itself performs the worst. This indicates that TextRank is less competitive to measure word trigger power since it does not take global information into consideration.

Weighting	Precision	Recall	F-measure
TF-IRF_w	0.368	0.452	0.355 ± 0.002
TextRank	0.345	0.424	0.332 ± 0.002
Product	0.368	0.451	0.354 ± 0.002

Table 7: Evaluation results for different methods for computing word trigger powers when $M = 3$ on the BOOK dataset.

Training Data Size We investigate the influence of training data size for social tag suggestion. As shown in Fig. 5, we increased the training data size from $8,000$ to $56,000$ step by $8,000$, and carried out evaluation on $4,000$ resources. The figure shows that:

- When the training data size is small (e.g., $8,000$), WTM can still achieve good suggestion performance.
- As the training data size increases, the performance of WTM improves, while the improvement speed declines.

The observation indicates that WTM does not require huge-size dataset to achieve good performance.

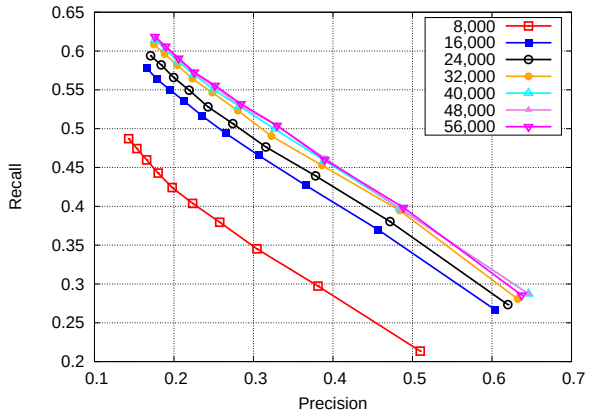


Figure 5: Precision-recall curves when the training data size increases from $8,000$ thousand to $56,000$ thousand on the BOOK dataset.

Conclusion By analyzing the influences of parameters on WTM, we find that WTM is robust to parameter variations.

4.4 Performance of EWTM

At the end of this section, we investigate the performance of EWTM for social tag suggestion. Here we simply set the smooth factor $\gamma = 0.5$.

As shown in Table 8, EWTM improves the performance of WTM (in Table 7) on the BOOK dataset when using TF-IRF$_w$ and the product as the methods for computing the word trigger powers, but decays when using TextRank. This verifies that TF-IRF$_w$ is the best method to measure word trigger powers for WTM. Table 8 indicates that emphasizing the tags appearing in the descriptions may enhance the suggestion power of the word trigger method.

Weighting	Precision	Recall	F-measure
TF-IRF$_w$	0.385	0.472	0.371 ± 0.001
TextRank	0.344	0.423	0.332 ± 0.002
Product	0.374	0.457	0.360 ± 0.001

Table 8: The evaluation results of EWTM with different methods for computing word trigger powers when $M = 3$ on the BOOK dataset.

However, the performance of EWTM on the BIBTEX dataset decays much compared to WTM. The F-measure of EWTM is only $F = 0.229$ compared with WTM $F = 0.267$. The main reason of the decay is that: the resource descriptions in the BIBTEX dataset are usually too short to provide sufficient information to precisely emphasize tags. In this case, EWTM may emphasize wrong tags and drop correct tags.

The experimental results on EWTM suggest that, the performance of EWTM is heavily influenced by the length of resource descriptions. Therefore, we have to analyze the characteristics of social tagging systems to decide whether to emphasize the tags that appear in the corresponding resource descriptions.

As future work, we will investigate the influence of the smooth factor γ to EWTM. It is also worth to investigate the problem when combining with collaboration-based methods for social tag suggestion.

5 Conclusion and Future Work

In this paper, we present a new perspective to social tagging and propose the word trigger method for social tag suggestion based on word alignment in statistical machine translation. Experiments show that our method is effective and efficient for social tag suggestion compared to other baselines.

There are still several open problems that should be further investigated:

1. We can exploit other word alignment methods like log-linear models (Liu et al., 2010a) for social tag suggestion.

2. We will ensemble WTM with other content-based and collaboration-based methods to build a practical social tag suggestion system.

3. WTM and EWTM can only suggest the tags that have appeared in translation models. In future, we plan to incorporate keyphrase extraction in social tag suggestion to make it suggest more appropriate tags not only from translation models but also from the resource descriptions.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 60873174. The authors would like to thank Peng Li for his insightful suggestions and thank the anonymous reviewers for their helpful comments.

References

M. Banko, V.O. Mittal, and M.J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of ACL*, pages 318–325.

A. Berger and J. Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of SIGIR*, pages 222–229.

A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. 2000. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of SIGIR*, pages 192–199.

D.M. Blei and M.I. Jordan. 2003. Modeling annotated data. In *Proceedings of SIGIR*, pages 127–134.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *JMLR*, 3:993–1022.

D.M. Blei, T.L. Griffiths, and M.I. Jordan. 2010. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):7.

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

M. Bundschus, S. Yu, V. Tresp, A. Rettinger, M. Dejori, and H.P. Kriegel. 2009. Hierarchical bayesian models for collaborative tagging systems. In *Proceedings of ICDM*, pages 728–733.

N. Dalvi, R. Kumar, B. Pang, and A. Tomkins. 2009. A translation model for matching reviews to objects. In *Proceeding of CIKM*, pages 167–176.

P. Duygulu, K. Barnard, J. De Freitas, and D. Forsyth. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. *Proceedings of ECCV*, pages 97–112.

A. Echihabi and D. Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of ACL*, pages 16–23.

D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. 2007. Automatic generation of social tags for music recommendation. In *Proceedings of NIPS*, pages 385–392.

E. Frank, G.W. Paynter, I.H. Witten, C. Gutwin, and C.G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of IJCAI*, pages 668–673.

S. Fujimura, KO Fujimura, and H. Okuda. 2008. Blogosonomy: Autotagging any text using bloggers' knowledge. In *Proceedings of WI*, pages 205–212.

J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of SIGIR*, pages 230–237.

J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53.

P. Heymann, D. Ramage, and H. Garcia-Molina. 2008. Social tag prediction. In *Proceedings of SIGIR*, pages 531–538.

A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme. 2006. Trend detection in folksonomies. *Semantic Multimedia*, pages 56–70.

T. Iwata, T. Yamada, and N. Ueda. 2009. Modeling social annotation data with content relevance using a topic model. In *Proceedings of NIPS*, pages 835–843.

R. Jaschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. 2008. Tag recommendations in social bookmarking systems. *AI Communications*, 21(4):231–247.

J. Jeon, V. Lavrenko, and R. Manmatha. 2003. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of SIGIR*, pages 119–126.

M. Karimzadehgan and C.X. Zhai. 2010. Estimation of statistical translation models based on mutual information for ad hoc information retrieval. In *Proceedings of SIGIR*, pages 323–330.

I. Katakis, G. Tsoumakas, and I. Vlahavas. 2008. Multilabel text classification for automated tag suggestion. *ECML PKDD Discovery Challenge 2008*, page 75.

R. Krestel, P. Fankhauser, and W. Nejdl. 2009. Latent dirichlet allocation for tag recommendation. In *Proceedings of ACM RecSys*, pages 61–68.

S.O.K. Lee and A.H.W. Chun. 2007. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ann semantic structures. In *Proceedings of WSEAS*, pages 88–93.

Z. Liu, P. Li, Y. Zheng, and M. Sun. 2009a. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*, pages 257–266.

Z. Liu, H. Wang, H. Wu, and S. Li. 2009b. Collocation extraction using monolingual word alignment method. In *Proceedings of EMNLP*, pages 487–495.

Y. Liu, Q. Liu, and S. Lin. 2010a. Discriminative word alignment by linear modeling. *Computational Linguistics*, 36(3):303–339.

Z. Liu, W. Huang, Y. Zheng, and M. Sun. 2010b. Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP*, pages 366–376.

Z. Liu, H. Wang, H. Wu, and S. Li. 2010c. Improving statistical machine translation with monolingual collocation. In *Proceedings of ACL*, pages 825–833.

Z. Liu, X. Chen, Y. Zheng, and M. Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of CoNLL*, pages 135–144.

C.D. Manning, P. Raghavan, and H. Schtze. 2008. *Introduction to information retrieval*. Cambridge University Press New York, NY, USA.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411.

R. Mirizzi, A. Ragone, T. Di Noia, and E. Di Sciascio. 2010. Semantic tags generation and retrieval for online advertising. In *Proceedings of CIKM*, pages 1089–1098.

G. Mishne. 2006. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *Proceedings of WWW*, pages 953–954.

V. Murdock and W.B. Croft. 2004. Simple translation models for sentence retrieval in factoid question answering. In *Proceedings of SIGIR*.

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

T. Ohkura, Y. Kiyota, and H. Nakagawa. 2006. Browsing system for weblog articles based on automated folksonomy. In *Proceedings of WWW*.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report, Stanford Digital Library Technologies Project*.

C. Quirk, C. Brockett, and W. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*, volume 149.

S. Ravi, A. Broder, E. Gabrilovich, V. Josifovski, S. Pandey, and B. Pang. 2010. Automatic generation of bid phrases for online advertising. In *Proceedings of WSDM*, pages 341–350.

S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. 2009. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of KDD*, pages 727–736.

P. Resnick and H.R. Varian. 1997. Recommender systems. *Communications of the ACM*, 40(3):56–58.

S. Riezler and Y. Liu. 2010. Query rewriting using monolingual statistical machine translation. *Computational Linguistics*, 36(3):569–582.

S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proccedings of ACL*, pages 464–471.

S. Riezler, Y. Liu, and A. Vasserman. 2008. Translating queries into snippets for improved query expansion. In *Proceedings of COLING*, pages 737–744.

X. Si and M. Sun. 2009. Tag-LDA for scalable real-time tag recommendation. *Journal of Computational Information Systems*, 6(1):23–31.

X. Si, Z. Liu, and M. Sun. 2010. Modeling social annotations via latent reason identification. *IEEE Intelligent Systems*, 25(6):42 – 49.

R. Soricut and E. Brill. 2006. Automatic question answering using the web: Beyond the factoid. *Information Retrieval*, 9(2):191–206.

M. Surdeanu, M. Ciaramita, and H. Zaragoza. 2008. Learning to rank answers on large online qa collections. In *Proceedings of ACL*, pages 719–727.

Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

P.D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336.

F.Y. Wang, K.M. Carley, D. Zeng, and W. Mao. 2007. Social computing: From social informatics to social intelligence. *IEEE Intelligent Systems*, 22(2):79–83.

R. Wetzker, C. Zimmermann, C. Bauckhage, and S. Albayrak. 2010. I tag, you tag: translating tags for advanced user models. In *Proceedings of WSDM*, pages 71–80.

X. Xue, J. Jeon, and W.B. Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of SIGIR*, pages 475–482.

Y. Yanbe, A. Jatowt, S. Nakamura, and K. Tanaka. 2007. Can social bookmarking enhance search in the web? In *Proceedings of JCDL*, pages 107–116.

S. Zhao, H. Wang, X. Lan, and T. Liu. 2010a. Leveraging multiple mt engines for paraphrase generation. In *Proceedings of COLING*, pages 1326–1334.

S. Zhao, H. Wang, and T. Liu. 2010b. Paraphrasing with search engine query logs. In *Proceedings of COLING*, pages 1317–1325.

T.C. Zhou, H. Ma, M.R. Lyu, and I. King. 2010. UserRec: A user recommendation framework in social tagging systems. In *Proceedings of AAAI*, pages 1486–1491.

Rumor has it: Identifying Misinformation in Microblogs

Vahed Qazvinian Emily Rosengren Dragomir R. Radev Qiaozhu Mei
University of Michigan
Ann Arbor, MI
{vahed,emirose,radev,qmei}@umich.edu

Abstract

A rumor is commonly defined as a statement whose true value is unverifiable. Rumors may spread misinformation (false information) or disinformation (deliberately false information) on a network of people. Identifying rumors is crucial in online social media where large amounts of information are easily spread across a large network by sources with unverified authority. In this paper, we address the problem of rumor detection in microblogs and explore the effectiveness of 3 categories of features: content-based, network-based, and microblog-specific memes for correctly identifying rumors. Moreover, we show how these features are also effective in identifying disinformers, users who endorse a rumor and further help it to spread. We perform our experiments on more than 10,000 manually annotated tweets collected from Twitter and show how our retrieval model achieves more than 0.95 in Mean Average Precision (MAP). Finally, we believe that our dataset is the first large-scale dataset on rumor detection. It can open new dimensions in analyzing online misinformation and other aspects of microblog conversations.

1 Introduction

A rumor is an unverified and instrumentally relevant statement of information spread among people (DiFonzo and Bordia, 2007). Social psychologists argue that rumors arise in contexts of ambiguity, when the meaning of a situation is not readily apparent, or potential threat, when people feel an acute need for security. For instance rumors about 'office renovation in a company' is an example of an ambiguous context, and the rumor that 'underarm deodorants cause breast cancer' is an example of a context in which one's well-being is at risk (DiFonzo et al., 1994).

The rapid growth of online social media has made it possible for rumors to spread more quickly. Online social media enable unreliable sources to spread large amounts of unverified information among people (Herman and Chomsky, 2002). Therefore, it is crucial to design systems that automatically detect misinformation and disinformation (the former often seen as simply false and the latter as deliberately false information).

Our definition of a rumor is established based on social psychology, where a rumor is defined as a statement whose truth-value is unverifiable or deliberately false. In-depth rumor analysis such as determining the intent and impact behind the spread of a rumor is a very challenging task and is not possible without first retrieving the complete set of social conversations (e.g., tweets) that are actually about the rumor. In our work, we take this first step to retrieve a complete set of tweets that discuss a specific rumor. In our approach, we address two basic problems. The first problem concerns retrieving online microblogs that are rumor-related. In the second problem, we try to identify tweets in which the rumor is endorsed (the posters show that they believe the rumor).

2 Related Work

We review related work on 3 main areas: Analyzing rumors, mining microblogs, and sentiment analysis and subjectivity detection.

2.1 Rumor Identification and Analysis

Though understanding rumors has been the subject of research in psychology for some time (Allport and Lepkin, 1945), (Allport and Postman, 1947), (DiFonzo and Bordia, 2007), research has

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1589–1599,
Edinburgh, Scotland, UK, July 27–31, 2011.

only recently begun to investigate how rumors are manifested and spread differently online. Microblogging services, like Twitter, allow small pieces of information to spread quickly to large audiences, allowing rumors to be created and spread in new ways (Ratkiewicz et al., 2010).

Related research has used different methods to study the spread of memes and false information on the web. Leskovec et al. use the evolution of quotes reproduced online to identify memes and track their spread overtime (Leskovec et al., 2009). Ratkiewicz et al. (Ratkiewicz et al., 2010) created the "Truthy" system, identifying misleading political memes on Twitter using tweet features, including hashtags, links, and mentions. Other projects focus on highlighting disputed claims on the Internet using pattern matching techniques (Ennals et al., 2010). Though our project builds on previous work, our work differs in its general focus on identifying rumors from a corpus of relevant phrases and our attempts to further discriminate between phrases that confirm, refute, question, and simply talk about rumors of interest.

Mendoza et al. explore Twitter data to analyze the behavior of Twitter users under the emergency situation of 2010 earthquake in Chile (Mendoza et al.,). They analyze the re-tweet network topology and find that the patterns of propagation in rumors differ from news because rumors tend to be questioned more than news by the Twitter community.

2.2 Sentiment Analysis

The automated detection of rumors is similar to traditional NLP sentiment analysis tasks. Previous work has used machine learning techniques to identify positive and negative movie reviews (Pang et al., 2002). Hassan et al. use a supervised Markov model, part of speech, and dependency patterns to identify attitudinal polarities in threads posted to Usenet discussion posts (Hassan et al., 2010). Others have designated sentiment scores for news stories and blog posts based on algorithmically generated lexicons of positive and negative words (Godbole et al., 2007). Pang and Lee provide a detailed overview of current techniques and practices in sentiment analysis and opinion mining (Pang and Lee, 2008; Pang and Lee, 2004).

Though rumor classification is closely related to opinion mining and sentiment analysis, it presents a different class of problem because we are concerned not just with the opinion of the person posting a tweet, but with whether the statements they post appear controversial. The automatic identification of rumors from a corpus is most closely related to the identification of memes done in (Leskovec et al., 2009), but presents new challenges since we seek to highlight a certain type of recurring phrases. Our work presents one of the first attempts at automatic rumor analysis.

2.3 Mining Twitter Data

With its nearly constant update of new posts and public API, Twitter can be a useful source for collecting data to be used in exploring a number of problems related to natural language processing and information diffusion (Bifet and Frank, 2010). Pak and Paroubek demonstrated experimentally that despite frequent occurrences of irregular speech patterns in tweets, Twitter can provide a useful corpus for sentiment analysis (Pak and Paroubek, 2010). The diversity of Twitter users make this corpus especially valuable. Ratkiewicz et al also use Twitter to detect and track misleading political memes (Ratkiewicz et al., 2010).

Along with many advantages, using Twitter as a corpus for sentiment analysis does present unusual challenges. Because posts are limited to 140 characters, tweets often contain information in an unusually compressed form and, as a result, grammar used may be unconventional. Instances of sarcasm and humor are also prevalent (Bifet and Frank, 2010). The procedures we used for the collection and analysis of tweets are similar to those described in previous work. However, our goal of developing computational methods to identify rumors being transmitted through tweets differentiates our project.

3 Problem Definition

Assume we have a set of tweets that are about the same topic that has some controversial aspects. Our objective in this work is two-fold: (1) Extract tweets that are about the controversial aspects of the story and spread misinformation (**Rumor retrieval**). (2) Identify users who believe that misinformation versus users who refute or question the rumor (**Belief**

Name	Rumor	Regular Expression Query	Status	#tweets
obama	Is Barack Obama muslim?	`Obama & (muslim|islam)`	false	4975
airfrance	Air France mid-air crash photos?	`(air.france|air france) & (photo|pic|pix)`	false	505
cellphone	Cell phone numbers going public?	`(cell|cellphone|cell phone)`	mostly false	215
michelle	Michelle Obama hired too many staff?	`staff & (michelle obama|first lady|1st lady)`	partly true	299
palin	Sarah Palin getting divorced?	`palin & divorce`	false	4423

Table 1: List of rumor examples and their corresponding queries used to collect data from Twitter

classification).

The following two tweets are two instances of the tweets written about president Obama and the Muslim world. The first tweet below is about president Obama and Muslim world, where the second tweet spread misinformation that president Obama is Muslim.

(non-rumor) "As Obama bows to Muslim leaders Americans are less safe not only at home but also overseas. Note: The terror alert in Europe... "

(rumor) "RT @johnnyA99 Ann Coulter Tells Larry King Why People Think Obama Is A Muslim http://bit.ly/9rs6pa #Hussein via @NewsBusters #tcot .."

The goal of the retrieval task is to discriminate between such tweets. In the second task, we use the tweets that are flagged as rumorous, and identify users that endorse (believe) the rumor versus users who deny or question it. The following three tweets are about the same story. The first user is a believer and the second and third are not.

(confirm) "RT @moronwatch: Obama's a Muslim. Or if he's not, he sure looks like one #whyimvotingrepublican."

(deny) "Barack Obama is a Christian man who had a Christian wedding with 2 kids baptised in Jesus name. Tea Party clowns call that muslim #p2 #gop"

(doubtful) "President Barack Obama's Religion: Christian, Muslim, or Agnostic? - The News of Today (Google): Share With Friend... http://bit.ly/bk42ZQ"

The first task is substantially more challenging than a standard IR task because of the requirement of both high precision (every result should be actually discussing the rumor) and high recall (the set should be complete). To do this, we submit a handcrafted regexp (extracted from about.com) to Twitter and retrieve a large primitive set of tweets that is supposed to have a high recall. This set however, contains a lot of false positives, tweets that match the regexp but are not about the rumor (e.g., "Obama meets muslim leaders"). Moreover, a rumor is usually stated using various instances (e.g., "Barack HUSSEIN Obama" versus "Obama is muslim"). Our goal is then to design a learning framework that filters all such false positives and retrieves various instances of the same rumor

Although our second task, belief classification, can be viewed as an opinion mining task, it is substantially different from opinion mining in nature. The difference from a standard opinion mining task is that here we are looking for attitudes about a subtle statement (e.g., "Palin is getting divorce") instead of the overall sentiment of the text or the opinion towards an explicit object or person (e.g., "Sarah Palin").

4 Data

As September 2010, Twitter reports that its users publish nearly 95 million tweets per day[1]. This makes Twitter an excellent case to analyze misinformation in social media.

Our goal in this work was to collect and annotate a large dataset that includes all the tweets that are written about a rumor in a certain period of time. To collect such a complete and self-contained dataset about a rumor, we used the Twitter search API, and retrieved all the tweets that matched a given regular expression. This API is the only API that returns results from the entire public Twitter stream and not a small randomly selected sample. To overcome the rate limit enforced by Twitter, we collected matching tweets once per hour, and remove any duplicates.

To use the search API, we carefully designed regular expression queries to be broad enough to match

[1] http://twitter.com/about

all the tweets that are about a rumor. Each query represents a popular rumor that is listed as "false" or only "partly true" on About.com's Urban Legends reference site[2] between 2009 and 2010. Table 1 lists the rumor examples that we used to collect our dataset along with their corresponding regular expression queries and the number of tweets collected.

4.1 Annotation

We asked two annotators to go over all the tweets in the dataset and mark each tweet with a "1" if it is about any of the rumors from Table 1, and with a "0" otherwise. This annotation scheme will be used in our first task to detect false positives, tweets that match the broad regular expressions and are retrieved, but are not about the rumor. For instance, both of the following tweets match the regular expression for the `palin` example, but only the second one is rumorous.

(0) "McCain Divorces Palin over her 'untruths and out right lies' in the book written for her. McCain's team says Palin is a petty liar and phony"

(1) "Sarah and Todd Palin to divorce, according to local Alaska paper. http://ow.ly/iNxF"

We also asked the annotators to mark each previously annotated rumorous tweet with "11" if the tweet poster endorses the rumor and with "12" if the user refutes the rumor, questions its credibility, or is neutral.

(12) "Sarah Palin Divorce Rumor Debunked on Facebook http://ff.im/62Evd"

(11) "Todd and Sarah Palin to divorce http://bit.ly/15StNc"

Our annotation of more than 10,400 tweets shows that $\%35$ of all the instances that matched the regular expressions are false positives, tweets that are not rumor-related but match the initial queries. Moreover, among tweets that are about particular rumors, nearly $\%43$ show the poster believe the rumor, demonstrating the importance of identifying misinformation and those who are misinformed. Table 2 shows the basic statistics extracted from the annotations for each story.

[2]http://urbanlegends.about.com

Rumor	non-rumor (0)	believe (11)	deny/ (12) doubtful/neutral	total
obama	3,036	926	1,013	4975
airfrance	306	71	128	505
cellphone	132	74	9	215
michelle	83	191	25	299
palin	86	1,709	2,628	4,423
total	3,643	2,971	3,803	10,417

Table 2: Number of instances in each class from the annotated data

task	κ
rumor retrieval	0.954
belief classification	0.853

Table 3: Inter-judge agreement in two annotation tasks in terms of κ-statistic

4.2 Inter-Judge Agreement

To calculate the annotation accuracy, we annotated 500 instances twice. These annotations were compared with each other, and the Kappa coefficient (κ) was calculated. The κ statistic is formulated as

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}$$

where $Pr(a)$ is the relative observed agreement among raters, and $Pr(e)$ is the probability that annotators agree by chance if each annotator is randomly assigning categories (Krippendorff, 1980; Carletta, 1996). Table 3 shows that annotators can reach a high agreement in both extracting rumors ($\kappa = 0.95$) and identifying believers ($\kappa = 0.85$).

5 Approach

In this section, we describe a general framework, which given a tweet, predicts (1) whether it is a rumor-related statement, and if so (2) whether the user believes the rumor or not. We describe 3 sets of features, and explain why these are intuitive to use for identification of rumors.

We process the tweets as they appear in the user timeline, and do not perform any pre-processing. Specially, we think that capitalization might be an important property. So, we do not lower-case the tweet texts either.

Our approach is based on building different Bayes classifiers as high level features and then learning a linear function of these classifiers for retrieval in the first task and classification in the second. Each

Bayes classifier, which corresponds to a feature f_i, calculates the likelihood ratio for a given tweet t, as shown in Equation 1.

$$\frac{P(\theta_i^+|t)}{P(\theta_i^-|t)} = \frac{P(\theta_i^+)}{P(\theta_i^-)} \frac{P(t|\theta_i^+)}{P(t|\theta_i^-)} \quad (1)$$

Here θ_i^+ and θ_i^- are two probabilistic models built based on feature f_i using a set of positive (+) and negative (−) training data. The likelihood ratio expresses how many times more likely the tweet t is under the positive model than the negative model with respect to f_i.

For computational reasons and to avoid dealing with very small numbers we use the log of the likelihood ratio to build each classifier.

$$LL_i = \log \frac{P(\theta_i^+|t)}{P(\theta_i^-|t)} = \log \frac{P(\theta_i^+)}{P(\theta_i^-)} + \log \frac{P(t|\theta_i^+)}{P(t|\theta_i^-)} \quad (2)$$

The first term $\frac{P(\theta_i^+)}{P(\theta_i^-)}$ can be easily calculated using the maximum likelihood estimates of the probabilities (i.e., the estimate of each probability is the corresponding relative frequency). The second term is calculated using various features that we explain below.

5.1 Content-based Features

The first set of features are extracted from the text of the tweets. We propose 4 content based features. We follow (Hassan et al., 2010) and present the tweet with 2 different patterns:

- Lexical patterns: All the words and segments in the tweet are represented as they appear and are tokenized using the space character.

- Part-of-speech patterns: All words are replaced with their part-of-speech tags. To find the part-of-speech of a hashtag we treat it as a word (since they could have semantic roles in the sentence), by omitting the tag sign, and then precede the tag with the label `TAG/`. We also introduce a new tag, `URL`, for URLs that appear in a tweet.

From each tweet we extract 4 (2×2) features, corresponding to unigrams and bigrams of each representation. Each feature is the log-likelihood ratio calculated using Equation 2. More formally, we represent each tweet t, of length n, lexically as $(w_1 w_2 \cdots w_n)$ and with part-of-speech tags as $(p_1 p_2 \cdots p_n)$. After building the positive and negative models (θ^+, θ^-) for each feature using the training data, we calculate the likelihood ratio as defined in Equation 2 where

$$\frac{P(t|\theta^+)}{P(t|\theta^-)} = \sum_{j=1}^{n} \log \frac{P(w_j|\theta^+)}{P(w_j|\theta^-)} \quad (3)$$

for unigram-lexical features (**`TXT1`**) and

$$\frac{P(t|\theta^+)}{P(t|\theta^-)} = \sum_{j=1}^{n-1} \log \frac{P(w_j w_{j+1}|\theta^+)}{P(w_j w_{j+1}|\theta^-)} \quad (4)$$

for bigram-based lexical features (**`TXT2`**). Similarly, we define the unigram and bigram-based part-of-speech features (**`POS1`** and **`POS2`**) as the log-likelihood ratio with respect to the positive and negative part-of-speech models.

5.2 Network-based Features

The features that we have proposed so far are all based on the content of individual tweets. In the second set of features we focus on user behavior on Twitter. We observe 4 types of network-based properties, and build 2 features that capture them.

Twitter enables users to re-tweet messages from other people. This interaction is usually easy to detect because the re-tweeted messages generally start with the specific pattern: '`RT @user`'. We use this property to infer about the re-tweeted message.

Let's suppose a user u_i re-tweets a message t from the user u_j (u_i: "`RT` @u_j t"). Intuitively, t is more likely to be a rumor if (1) u_j has a history of posting or re-tweeting rumors, or (2) u_i has posted or re-tweeted rumors in the past.

Given a set of training instances, we build a positive (θ^+) and a negative (θ^-) user models. The first model is a probability distribution over all users that have posted a positive instance or have been re-tweeted in a positive instance. Similarly, the second model is a probability distribution over users

that have posted (or been re-tweeted in) a negative instance. After building the models, for a given tweet we calculate two log-likelihood ratios as two network-based features.

The first feature is the log-likelihood ratio that u_i is under a positive user model (**USR1**) and the second feature is the log-likelihood ratio that the tweet is re-tweeted from a user (u_j) who is under a positive user model than a negative user model (**USR2**).

The distinction between the posting user and the re-tweeted user is important, since some times the users modify the re-tweeted message in a way that changes its meaning and intent. In the following example, the original user is quoting president Obama. The second user is re-tweeting the first user, but has added more content to the tweet and made it sound rumorous.

original message (non-rumor) "Obama says he's doing 'Christ's work'."

re-tweeted (rumor) "Obama says he's doing 'Christ's work.' Oh my God, CHRIST IS A MUSLIM."

5.3 Twitter Specific Memes

Our final set of features are extracted from memes that are specific to Twitter: hashtags and URLs. Previous work has shown the usefulness of these memes (Ratkiewicz et al., 2010).

5.3.1 Hashtags

One emergent phenomenon in the Twitter ecosystem is the use of hashtags: words or phrases prefixed with a hash symbol (#). These hashtags are created by users, and are widely used for a few days, then disappear when the topic is outdated (Huang et al., 2010).

In our approach, we investigate whether hashtags used in rumor-related tweets are different from other tweets. Moreover, we examine whether people who believe and spread rumors use hashtags that are different from those seen in tweets that deny or question a rumor.

Given a set of training tweets of positive and negative examples, we build two statistical models (θ^+, θ^-), each showing the usage probability distribution of various hashtags. For a given tweet, t, with a set of m hashtags ($\#h_1 \cdots \#h_m$), we calculate the log-likelihood ratio using Equation 2 where

	Feature	LL-ratio	model
Content	**TXT1**	content unigram	content unigram
	TXT2	content bigram	content unigram
	POS1	content pos	content pos unigram
	POS2	content pos	content pos bigram
Twitter	**URL1**	content unigram	target URL unigram
	URL2	content bigram	target URL bigram
	TAG	hashtag	hashtag
Network	**USR1**	tweeting user	all users in the data
	USR2	re-tweeted user	all users in the data

Table 4: List of features used in our optimization framework. Each feature is a log-likelihood ratio calculated against a a positive (+) and negative (−) training models.

$$\frac{P(t|\theta^+)}{P(t|\theta^-)} = \sum_{j=1}^{m} \log \frac{P(\#h_j|\theta^+)}{P(\#h_j|\theta^-)} \quad (5)$$

5.3.2 URLs

Previous work has discussed the role of URLs in information diffusion on Twitter (Honeycutt and Herring, 2009). Twitter users share URLs in their tweets to refer to external sources or overcome the length limit forced by Twitter. Intuitively, if a tweet is a positive instance, then it is likely to be similar to the content of URLs shared by other positive tweets. Using the same reasoning, if a tweet is a negative instance, then it should be more similar to the web pages shared by other negative instances.

Given a set of training tweets, we fetch all the URLs in these tweets and build θ^+ and θ^- once for unigrams and once for bigrams. These models are merely built on the content of the URLs and ignore the tweet content. Similar to previous features, we calculate the log-likelihood ratio of the content of each tweet with respect to θ^+ and θ^- for unigrams (**URL1**) and bigrams **URL2**).

Table 4 summarizes the set of features used in our proposed framework, where each feature is a log-likelihood ratio calculated against a positive (+) and negative (−) training models. To build these language models, we use the CMU Language Modeling toolkit (Clarkson and Rosenfeld, 1997).

5.4 Optimization

We build an L_1-regularized log-linear model (Andrew and Gao, 2007) on various features discussed before to predict each tweet. Suppose, a procedure generates a set of candidates for an input x. Also,

let's suppose $\Phi : X \times Y \rightarrow \mathbb{R}^D$ is a function that maps each (x, y) to a vector of feature values. Here, the feature vector is the vector of coefficients corresponding to different network, content, and twitter-based properties, and the parameter vector $\theta \in \mathbb{R}^D$ ($D \leq 9$ in our experiments) assigns a real-valued weight to each feature. This estimator chooses θ to minimize the sum of least squares and a regularization term R.

$$\hat{\theta} = arg\min_{\theta}\{\frac{1}{2}\sum_{i}||\langle\theta, x_i\rangle - y_i||_2^2 + R(\theta)\} \quad (6)$$

where the regularizer term $R(\theta)$ is the weighted L_1 norm of the parameters.

$$R(\theta) = \alpha\sum_{j}|\theta_j| \quad (7)$$

Here, α is a parameter that controls the amount of regularization (set to 0.1 in our experiments).

Gao et. al (Gao et al., 2007) argue that optimizing L_1-regularized objective function is challenging since its gradient is discontinuous whenever some parameters equal zero. In this work, we use the *orthant-wise limited-memory quasi-Newton* algorithm (OWL-QN), which is a modification of L-BFGS that allows it to effectively handle the discontinuity of the gradient (Andrew and Gao, 2007). OWL-QN is based on the fact that when restricted to a single orthant, the L_1 regularizer is differentiable, and is in fact a linear function of θ. Thus, as long as each coordinate of any two consecutive search points does not pass through zero $R(\theta)$ does not contribute at all to the curvature of the function on the segment joining them. Therefore, we can use L-BFGS to approximate the Hessian of $L(\theta)$ alone and use it to build an approximation to the full regularized objective that is valid on a given orthant. This algorithm works quite well in practice, and typically reaches convergence in even fewer iterations than standard L-BFGS (Gao et al., 2007).

6 Experiments

We design 2 sets of experiments to evaluate our approach. In the first experiment we assess the effectiveness of the proposed method when employed in an Information Retrieval (IR) framework for rumor retrieval and in the second experiment we employ various features to detect users' beliefs in rumors.

6.1 Rumor Retrieval

In this experiment, we view different stories as queries, and build a relevance set for each query. Each relevance set is an annotation of the entire 10,417 tweets, where each tweet is marked as *relevant* if it matches the regular expression query and is marked as a rumor-related tweet by the annotators. For instance, according to Table 2 the `cellphone` dataset has only 83 relevant documents out of the entire 10,417 documents.

For each query we use 5-fold cross-validation, and predict the relevance of tweets as a function of their features. We use these predictions and rank all the tweets with respect to the query. To evaluate the performance of our ranking model for a single query (Q) with the set of relevant documents $\{d_1, \cdots, d_m\}$, we calculate Average Precision as

$$AP(Q) = \frac{1}{m}\sum_{k=1}^{m}\text{Precision}(R_k) \quad (8)$$

where R_k is the set of ranked retrieval results from the top result to the k^{th} relevant document, d_k (Manning et al., 2008).

6.1.1 Baselines

We compare our proposed ranking model with a number of other retrieval models. The first two simple baselines that indicate a difficulty lower-bound for the problem are **Random** and **Uniform** methods. In the Random baseline, documents are ranked based on a random number assignment to them. In the Uniform model, we use a 5-fold cross validation, and in each fold the label of the test documents is determined by the majority vote from the training set.

The main baseline that we use in this work, is the regular expression that was submitted to Twitter to collect data (**regexp**). Using the same regular expression to mark the relevance of the documents will cause a recall value of 1.00 (since it will retrieve all the relevant documents), but will also retrieve false positives, tweets that match the regular expression but are not rumor-related. We would like to investigate whether using training data will help us decrease the rate of false positives in retrieval.

Finally, using the Lemur Toolkit software[3], we employ a KL divergence retrieval model with

[3]http://www.lemurproject.org/

Dirichlet smoothing (**KL**). In this model, documents are ranked according to the negation of the divergence of query and document language models. More formally, given the query language model θ_Q, and the document language model θ_D, the documents are ranked by $-D(\theta_Q||\theta_D)$, where D is the KL-divergence between the two models.

$$D(\theta_Q||\theta_D) = \sum_w p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_D)} \quad (9)$$

To estimate $p(w|\theta_D)$, we use Bayesian smoothing with Dirichlet priors (Berger, 1985).

$$p_s(w|\theta_D) = \frac{C(w,D) + \mu.p(w|\theta_S)}{\mu + \sum_w C(w,D)} \quad (10)$$

where, μ is a parameter, C is the count function, and $theta_S$ is the collection language model. Higher values of μ put more emphasis on the collection model. Here, we try two variants of the model, one using the default parameter value in Lemur ($\mu = 2000$), and one in which μ is tuned based on the the data ($\mu = 10$). Using the test data to tune the parameter value, μ, will help us find an upper-bound estimate of the effectiveness of this method.

Table 5 shows the Mean Average Precision (MAP) and $F_{\beta=1}$ for each method in the rumor retrieval task. This table shows that a method that employs training data to re-rank documents with respect to rumors makes significant improvements over the baselines and outperforms other strong retrieval systems.

6.1.2 Feature Analysis

To investigate the effectiveness of using individual features in retrieving rumors, we perform 5-fold cross validations for each query, using different feature sets each time. Figure 1 shows the average precision and recall for our proposed optimization system when content-based (**TXT1**+**TXT2**+**POS1**+**POS2**), network-based (**USR1**+**USR2**), and twitter specific memes (**TAG**+**URL1**+**URL2**) are employed individually.

Figure 1 shows that features that are calculated using the content language models are very effective in achieving high precision and recall. Twitter specific features, especially hashtags, can result in high precisions but lead to a low recall value because many tweets do not share hashtags or are not written based on the contents of external URLs.

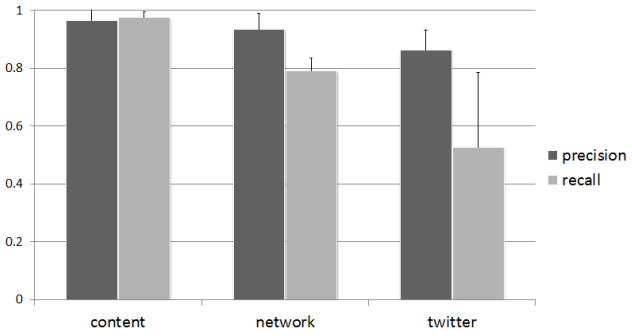


Figure 1: Average precision and recall of the proposed method employing each set of features: content-based, network-based, and twitter specific.

Finally, we find that user history can be a good indicator of rumors. However, we believe that this feature could be more helpful with a complete user set and a more comprehensive history of their activities.

6.1.3 Domain Training Data

As our last experiment with rumor retrieval we investigate how much new labeled data from an emergent rumor is required to effectively retrieve instances of that particular rumor. This experiment helps us understand how our proposed framework could be generalized to other stories.

To do this experiment, we use the `obama` story, which is a large dataset with a significant number of false positive instances. We extract 400 randomly selected tweets from this dataset and keep them for testing. We also build an initial training dataset of the other 4 rumors, and label them as *not relevant*. We assess the performance of the retrieval model as we gradually add the rest of the `obama` tweets. Figure 2 shows both Average Precision and labeling accuracy versus the size of the labeled data used from the `obama` dataset. This plot shows that both measures exhibit a fast growth and reach 80% when the number of labeled data reaches 2000.

6.2 Belief Classification

In previous experiments we showed that maximizing a linear function of log-likelihood ratios is an effective method in retrieving rumors. Here, we in-

Method	MAP	95% C.I.	$F_{\beta=1}$	95% C.I.
Random	0.129	[-0.065, 0.323]	0.164	[-0.051, 0.379]
Uniform	0.129	[-0.066, 0.324]	0.198	[-0.080, 0.476]
regexp	0.587	[0.305, 0.869]	0.702	[0.479, 0.925]
KL ($\mu = 2000$)	0.678	[0.458, 0.898]	0.538	[0.248, 0.828]
KL ($\mu = 10$)	0.803	[0.641, 0.965]	0.681	[0.614, 0.748]
LL (all 9 features)	**0.965**	[0.936, 0.994]	**0.897**	[0.828, 0.966]

Table 5: Mean Average Precision (MAP) and $F_{\beta=1}$ of each method in the rumor retrieval task. (C.I.: Confidence Interval)

Method	Accuracy	Precision	Recall	$F_{\beta=1}$	Win/Loss Ratio
random	0.501	0.441	0.513	0.474	1.004
uniform	0.439	0.439	1.000	0.610	0.781
TXT	0.934	0.925	0.924	0.924	14.087
POS	0.742	0.706	0.706	0.706	2.873
content (**TXT+POS**)	**0.941**	0.934	0.930	**0.932**	15.892
network (**USR**)	0.848	0.873	0.765	0.815	5.583
TAG	0.589	0.734	0.099	0.175	1.434
URL	0.664	0.630	0.570	0.598	1.978
twitter (**TAG+URL**)	0.683	0.658	0.579	0.616	2.155
all	0.935	**0.944**	0.906	0.925	14.395

Table 6: Accuracy, precision, recall, $F_{\beta=1}$, and win/loss ratio of belief classification using different features.

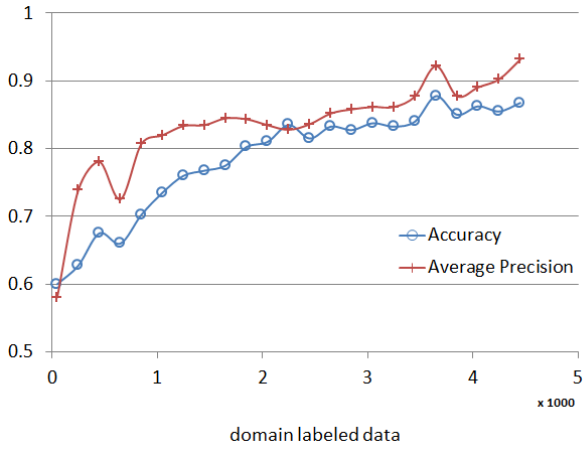


Figure 2: Average Precision and Accuracy learning curve for the proposed method employing all 9 features.

vestigate whether this method, and in particular, the proposed features are useful in detecting users' beliefs in a rumor that they post about. Unlike retrieval, detecting whether a user endorses a rumor or refutes it may be possible using similar methods regardless of the rumor. Intuitively, linguistic features such as negation (e.g., "obama is not a muslim"), or capitalization (e.g., "barack HUSSEIN obama ..."), user history (e.g., liberal tweeter vs. conservative tweeter), hashtags (e.g., #tcot vs. #tdot), and URLs (e.g., links to fake airfrance crash photos) should help to identify endorsements.

We perform this experiment by making a pool of all the tweets that are marked as "rumorous" in the annotation task. Table 2 shows that there are 6,774 such tweets, from which 2,971 show belief and 3,803 tweets show that the user is doubtful, denies, or questions it.

Using various feature settings, we perform 5-fold cross-validation on these 6,774 rumorous tweets. Table 6 shows the results of this experiment in terms of F-score, classification accuracy, and win/loss ratio, the ratio of correct classification to an incorrect

classification.

7 Conclusion

In this paper we tackle the fairly unaddressed problem of identifying misinformation and disinformers in Microblogs. Our contributions in this paper are two-fold: (1) We propose a general framework that employs statistical models and maximizes a linear function of log-likelihood ratios to retrieve rumorous tweets that match a more general query. (2) We show the effectiveness of the proposed feature in capturing tweets that show user endorsement. This will help us identify disinformers or users that spread false information in online social media.

Our work has resulted in a manually annotated dataset of 10,000 tweets from 5 different controversial topics. To the knowledge of authors this is the first large-scale publicly available rumor dataset, and can open many new dimensions in studying the effects of misinformation or other aspects of information diffusion in online social media.

In this paper we effectively retrieve instances of rumors that are already identified and evaluated by an external source such as About.com's Urban Legends reference. Identifying new emergent rumors directly from the Twitter data is a more challenging task. As our future work, we aim to build a system that employs our findings in this paper and the emergent patterns in the re-tweet network topology to identify whether a new trending topic is a rumor or not.

8 Acknowledgments

The authors would like to thank Paul Resnick, Rahul Sami, and Brendan Nyhan for helpful discussions. This work is supported by the National Science Foundation grant "SoCS: Assessing Information Credibility Without Authoritative Sources" as IIS-0968489. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the supporters.

References

Floyd H. Allport and Milton Lepkin. 1945. Wartime rumors of waste and special privilege: why some people believe them. *Journal of Abnormal and Social Psychology*, 40(1):3 – 36.

Gordon Allport and Leo Postman. 1947. *The psychology of rumor*. Holt, Rinehart, and Winston, New York.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of l1-regularized log-linear models. In *ICML '07*, pages 33–40.

James Berger. 1985. *Statistical decision theory and Bayesian Analysis (2nd ed.)*. New York: Springer-Verlag.

Albert Bifet and Eibe Frank. 2010. Sentiment knowledge discovery in twitter streaming data. In Bernhard Pfahringer, Geoff Holmes, and Achim Hoffmann, editors, *Discovery Science*, volume 6332 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Comput. Linguist.*, 22(2):249–254.

Philip Clarkson and Roni Rosenfeld. 1997. Statistical language modeling using the cmu-cambridge toolkit. *Proceedings ESCA Eurospeech*, 47:45–148.

Nicholas DiFonzo and Prashant Bordia. 2007. Rumor, gossip, and urban legend. *Diogenes*, 54:19–35, February.

Nicholas DiFonzo, P. Prashant Bordia, and Ralph L. Rosnow. 1994. Reining in rumors. *Organizational Dynamics*, 23(1):47–62.

Rob Ennals, Dan Byler, John Mark Agosta, and Barbara Rosario. 2010. What is disputed on the web? In *Proceedings of the 4th workshop on Information Credibility*, WICOW '10, pages 67–74.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *ACL '07*.

Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, Boulder, CO, USA.

Ahmed Hassan, Vahed Qazvinian, and Dragomir Radev. 2010. What's with the attitude? identifying sentences with attitude in online discussions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1245–1255, Cambridge, MA, October. Association for Computational Linguistics.

Edward S Herman and Noam Chomsky. 2002. *Manufacturing Consent: The Political Economy of the Mass Media*. Pantheon.

Courtenay Honeycutt and Susan C. Herring. 2009. Beyond microblogging: Conversation and collaboration

via twitter. *Hawaii International Conference on System Sciences*, 0:1–10.

Jeff Huang, Katherine M. Thornton, and Efthimis N. Efthimiadis. 2010. Conversational tagging in twitter. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, HT '10, pages 173–178.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to its Methodology*. Beverly Hills: Sage Publications.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. Twitter under crisis: Can we trust what we rt?

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL'04*, Morristown, NJ, USA.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2:1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of conference on Empirical methods in natural language processing, EMNLP'02*, pages 79–86.

Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. 2010. Detecting and tracking the spread of astroturf memes in microblog streams. *CoRR*, abs/1011.3768.

Exploiting Parse Structures for Native Language Identification

Sze-Meng Jojo Wong
Centre for Language Technology
Macquarie University
Sydney, Australia
sze.wong@mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
Sydney, Australia
mark.dras@mq.edu.au

Abstract

Attempts to profile authors according to their characteristics extracted from textual data, including native language, have drawn attention in recent years, via various machine learning approaches utilising mostly lexical features. Drawing on the idea of contrastive analysis, which postulates that syntactic errors in a text are to some extent influenced by the native language of an author, this paper explores the usefulness of syntactic features for native language identification. We take two types of parse substructure as features—horizontal slices of trees, and the more general feature schemas from discriminative parse reranking—and show that using this kind of syntactic feature results in an accuracy score in classification of seven native languages of around 80%, an error reduction of more than 30%.

1 Introduction

Inferring characteristics of authors from their textual data, often termed *authorship profiling*, has seen a number of computational approaches proposed in recent years. The problem is typically treated as a classification task, where an author is classified with respect to characteristics such as gender, age, native language, and so on. This profile information is often of interest to marketing organisations for product promotional reasons as well as governments or law enforcements for crime investigation purposes. The particular application that motivates the present study is detection of phishing (Myers, 2007), the attempt to defraud through texts that are designed to deceive Internet users into giving away confidential details. One class of countermeasures to phishing consists of technical methods such as email authentication; another looks at profiling of the text's author(s) (Fette et al., 2007; Zheng et al., 2003), to find any indications of the source of the text.

In this paper we investigate classification of a text with respect to an author's native language, where this is not the language that that text is written in (which is often the case in phishing); we refer to this as *native language identification*. Initial work by Koppel et al. (2005) was followed by Tsur and Rappoport (2007), Estival et al. (2007), van Halteren (2008), and Wong and Dras (2009). By and large, the problem was tackled using various supervised machine learning approaches, with mostly lexical features over characters, words, and parts of speech, as well as some document structure.

Syntactic features, in contrast, in particular those that capture grammatical errors, which might potentially be useful for this task, have received little attention. Koppel et al. (2005) did suggest using syntactic errors in their work but did not investigate them in any detail. Wong and Dras (2009) noted the relevance of the concept of *contrastive analysis* (Lado, 1957), which postulates that native language constructions lead to characteristic errors in a second language. In their experimental work, however, they used only three manual syntactic constructions drawn from the literature; an ANOVA analysis showed a detectable effect, but they did not improve classification accuracy over purely lexical features.

In this paper, we investigate syntactic features for native language identification that are more general

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1600–1610,
Edinburgh, Scotland, UK, July 27–31, 2011.

than, and that do not require the manual construction of, the above approach. Taking the trees produced by statistical parsers, we use tree cross-sections as features in a machine learning approach to determine which ones characterise non-native speaker errors. Specifically, we look at two types of parse tree substructure to use as features: horizontal slices of the trees—that is, characterising parse trees as sets of context-free grammar production rules—and the features schemas used in discriminative parse reranking. The goal of the present study is therefore to investigate the influence to which syntactic features represented by parse structures would have on the classification task of identifying an author's native language relative to, and in combination with, lexical features.

The remainder of this paper is structured as follows. In Section 2, we discuss some related work on the two key topics of this paper: primarily on comparable work in native language identification, and then on how the notion of contrastive analysis can be applicable here. We then describe the models examined in Section 3, followed by experimental setup in Section 4. Section 5 presents results, and Section 6 discussion of those results.

2 Related Work

2.1 Native Language Identification

The earliest work on native language identification in this classification paradigm is that of Koppel et al. (2005), in which they deployed a machine learning approach to the task, using as features function words, character n-grams, and part-of-speech (PoS) bi-grams, as well as some spelling mistakes. With five different groups of English authors (of native languages Bulgarian, Czech, French, Russian, and Spanish) selected from the first version of *International Corpus of Learner English* (ICLE), they gained a relatively high classification accuracy of 80%. Koppel et al. (2005) also suggested that syntactic features (syntactic errors) might be useful features, but only investigated this idea at a shallow level by treating rare PoS bigrams as ungrammatical structures.

Tsur and Rappoport (2007) replicated the work of Koppel et al. (2005) to investigate the hypothesis that the choice of words in second language writing is highly influenced by the frequency of native language syllables — the *phonology* of the native language. Approximating this by character bi-grams alone, they managed to achieve a classification accuracy of 66%.

Native language is also amongst the characteristics investigated in the task of authorship profiling by Estival et al. (2007), as well as other demographic and personality characteristics. This study used a variety of lexical and document structure features. For the native language identification classification task, their model yielded a reasonably high accuracy of 84%, but this was over a set of only three languages (Arabic, English and Spanish) and against a most frequent baseline of 62.9%.

Another related work is that of van Halteren (2008), who used the Europarl corpus of parliamentary speeches. In Europarl, one original language is transcribed, and the others translated from it; the task was to identify the original language. On the basis of frequency counts of word-based n-grams, surprisingly high classification accuracies within the range of 87-97% were achieved across six languages (English, German, French, Dutch, Spanish, and Italian). This turns out, however, to be significantly influenced by the use of particular phrases used by speakers of different languages in the parliamentary context (e.g. the way Germans typically address the chamber).

To our knowledge, Wong and Dras (2009) is the only work that has investigated the usefulness of syntactic features for the task of native language identification. They first replicated the work of Koppel et al. (2005) with the three types of lexical feature, namely function words, character n-grams, and PoS bi-grams. They then examined the literature on contrastive analysis (see Section 2.2), from the field of second language acquisition, and selected three syntactic errors commonly observed in non-native English users—subject-verb disagreement, noun-number disagreement and misuse of determiners—that had been identified as being influenced by the native language. An ANOVA analysis showed that the native language identification constructions were identifiable; however, the overall classification was not improved over the lexical features by using just the three manually detected syntactic errors. The best overall accuracy re-

ported was 73.71%; this was on the second version of ICLE, across seven languages (those of Koppel et al. (2005), plus the two Asian languages Chinese and Japanese).

As a possible approach that would improve the classification accuracy over just the three manually detected syntactic errors, Wong and Dras (2009) suggested deploying (but did not carry out) an idea put forward by Gamon (2004) (citing Baayen et al. (1996)) for the related task of identifying the author of a text: to use CFG production rules to characterise syntactic structures used by authors.[1] We note that similar ideas have been used in the task of sentence grammaticality judgement, which utilise parser outputs (both trees and by-products) as classification features (Mutton et al., 2007; Sun et al., 2007; Foster et al., 2008; Wagner et al., 2009; Tetreault et al., 2010; Wong and Dras, 2010). We combine this idea with one we introduce in this paper, of using discriminative reranking features as a broader characterisation of the parse tree.

2.2 Contrastive analysis

Contrastive analysis (Lado, 1957) was an early attempt in the field of second language acquisition to explain the kinds and source of errors that non-native speakers make. It arose out of behaviourist psychology, and saw language learning as an issue of habit formation that could be inhibited by previous habits inculcated in learning the native language. The theory was also tied to structural linguistics: it compared the syntactic structures of the native and second languages to find differences that might cause learning difficulties. The Lado work postulated the Contrastive Analysis Hypothesis (CAH), claiming that "those elements which are similar to [the learner's] native language will be simple for him, and those elements that are different will be difficult"; the consequence is that there will be more errors made in those difficult elements.

While contrastive analysis was influential at first, it was increasingly noticed that many errors were common across all language learners regardless of native language, which could not be explained under contrastive analysis. Corder (1967) then described an alternative, *error analysis*, where contrastive analysis-style errors were seen as only one type of error, 'interlanguage' or 'interference' errors; other types were 'intralingual' and 'developmental' errors, which are not specific to the native language (Richards, 1971).

In an overview of contrastive analysis after the emergence of error analysis, Wardhaugh (1970) noted that there were two interpretations of the CAH, termed the strong and weak forms. Under the strong form, all errors were attributed to the native language, and clearly that was not tenable in light of error analysis evidence. In the weak form, these differences have an influence but are not the sole determinant of language learning difficulty. Wardhaugh noted claims at the time that the hypothesis was no longer useful in either the strong or the weak version: "Such a claim is perhaps unwarranted, but a period of quiescence is probable for CA itself". This appears to be the case, with the then-dominant error analysis giving way to newer, more specialised theories of second language acquisition, such as the *competition model* of MacWhinney and Bates (1989) or the *processability theory* of Pienemann (1998). Nevertheless, smaller studies specifically of interlanguage errors have continued to be carried out, generally restricted in their scope to a specific grammatical aspect of English in which the native language of the learners might have an influence. To give some examples, Granger and Tyson (1996) examined the usage of connectors in English by a number of different native speakers – French, German, Dutch, and Chinese; Vassileva (1998) investigated the employment of first person singular and plural by another different set of native speakers – German, French, Russian, and Bulgarian; Slabakova (2000) explored the acquisition of telicity marking in English by Spanish and Bulgarian learners; Yang and Huang (2004) studied the impact of the absence of grammatical tense in Chinese on the acquisition of English tense-aspect system (i.e. telicity marking); Franck et al. (2002) and Vigliocco et al. (1996) specifically examined the usage of subject-verb agreement in English by French and Spanish, respectively. There are also a few teaching resources

[1] It is not entirely clear how this might work for authorship identification: would the Brontë sisters, the corpus Gamon worked with, have used a significant number of different syntactic constructions from each other? In the context of native language identification, however, constrastive analysis postulates that this is exactly the case for the different classes.

for English language teachers that collate such phenomena, such as that of Swan and Smith (2001).

NLP techniques and a probabilistic view of native language identification now let us revisit and make use of the weak form of the CAH. Interlanguage errors, as represented by differences in parse trees, may be characteristic of the native language of a learner; we can use the occurrence of these to come up with a revised likelihood of the native language. In this paper, we use machine learning in a prediction task as our approach to this.

3 Models

This section describes the three basic models investigated: the lexical model, based on Koppel et al. (2005), as the baseline; and then the two models that exploit syntactic information. In Section 5 we look at the performance of each model independently and also in combination: to combine, we just concatenate feature vectors.

Lexical As Wong and Dras (2009), we replicate the features of Koppel et al. (2005) to produce our LEXICAL model. These are of three types: function words,[2] character n-grams, and PoS n-grams. We follow Wong and Dras (2009) in resolving some unclear issues from Koppel et al. (2005). Specifically, we use the same list of function words, left unspecified in Koppel et al. (2005), that were empirically determined by Wong and Dras (2009) to be the best of three candidates; we used character bi-grams, as the best performing n-grams, although this also had been left unspecified by Koppel et al. (2005); and we used the most frequently occurring PoS bi-grams and tri-grams, obtained by using the Brill tagger provided in NLTK (Bird et al., 2009) being trained on the Brown corpus. In total, there are 798 features of this class with 398 function words, 200 most frequently occurring character bi-grams, and 200 most frequently occurring PoS bi-grams. Both function words and PoS bi-grams have feature values of binary type; while for character bi-grams, the feature value is the relative frequency. (These types of feature value are the best performing one for each lexical feature.)

We omitted the 250 rare bi-grams used by Koppel et al. (2005), as an ablative analysis showed that they contributed nothing to classification accuracy.

Production Rules Under this model (PRODRULE), we take as features horizontal slices of parse trees, in effect treating them as sets of CFG production rules. Feature values are binary. We look at all possible rules as features, but also present results for subsets of features chosen using feature selection. For each language in our dataset, we identify the n rules most characteristic of the language using Information Gain (IG). For m classes, we use the formulation of Yang and Pedersen (1997):

$$\begin{aligned} IG(r) = & -\sum_{i=1}^{m} \Pr(c_i) \log \Pr(c_i) \\ & + \Pr(r) \sum_{i=1}^{m} \Pr(c_i|r) \log \Pr(c_i|r) \\ & + \Pr(\bar{r}) \sum_{i=1}^{m} \Pr(c_i|\bar{r}) \log \Pr(c_i|\bar{r}) \end{aligned} \quad (1)$$

We also investigated simple frequencies, frequency ratios, and pointwise mutual information; as in much other work, IG performed best, so we do not present results for the others. Bi-normal separation (Forman, 2003), often competitive with IG, is only suitable for binary classification.

It is worth noting that the production rules being used here are all non-lexicalised ones, except those lexicalised with function words and punctuation, to avoid topic-related clues.

Reranking Features As opposed to the horizontal parse production rules, features used for discriminative reranking are cross-sections of parse trees that might capture other aspects of ungrammatical structures. For these we use the 13 feature schemas described in Charniak and Johnson (2005), which were inspired by earlier work in discriminative estimation techniques, such as Johnson et al. (1999) and Collins (2000). Examples of these feature schemas include tuples covering head-to-head dependencies, preterminals together with their closest maximal projection ancestors, and subtrees rooted in the least common ancestor.

These feature schemas are not the only possible ones—they were empirically selected for the specific purpose of augmenting the Charniak parser. However, much subsequent work has tended to use

[2] As with most work in authorship profiling, only function words are used, so that the result is not tied to a particular domain, and no clues are obtained from different topics that different authors might write about.

these same features, albeit sometimes with extensions for specific purposes (e.g. Johnson and Ural (2010) for the Berkeley parser (Petrov et al., 2006), Ng et al. (2010) for the C&C parser (Clark and Curran, 2007)). We also use this standard set, specifically the set of instantiated feature schemas from the parser from Charniak and Johnson (2005) as trained on the Wall Street Journal (WSJ), which gives 1,333,837 potential features.

4 Experimental Setup

4.1 Data

We use the *International Corpus of Learner English* (ICLE) compiled by Granger et al. (2009) for the precise purpose of studying the English writings of non-native English learners from diverse countries. All the contributors to the corpus are claimed to possess similar English proficiency levels (ranging from intermediate to advanced learners) and are in the same age group (all in their twenties at the time of corpus collection.) This was also the data used by Koppel et al. (2005) and Tsur and Rappoport (2007), although where they used the first version of the corpus, we use version 2.

Briefly, the first version contains 11 sub-corpora of English essays contributed by second-year and third-year university students of different native language backgrounds (mostly European and Slavic languages) — Bulgarian, Czech, Dutch, Finnish, French, German, Italian, Polish, Russian, Spanish, and Swedish; the second version has been extended to additional 5 other native languages (including Asian languages) — Chinese, Japanese, Norwegian, Turkish, and Tswana.

As per Wong and Dras (2009), we examine seven languages, namely Bulgarian, Czech, French, Russian, Spanish, Chinese, and Japanese. For each native language, we randomly select from amongst essays with length of 500-1000 words. For the purpose of the present study, we have 95 essays per native language. For the same reason as highlighted by Wong and Dras (2009), we intentionally use fewer essays as compared to Koppel et al. (2005)[3] with a view to reserving more data for future work. We divide these into training sets of 70 essays per language, with a held-out test set of 25 essays per language. There are 17,718 training sentences and 6,791 testing sentences.

4.2 Parsers

We use two parsers: the Stanford parser (Klein and Manning, 2003) and the Charniak and Johnson (henceforth C&J) parser (Charniak and Johnson, 2005). Both are widely used, and produce relatively accurate parses: the Stanford parser gets a labelled f-score of 85.61 on the WSJ, and the C&J 91.09.

With the Stanford parser, there are 26,284 unique parse production rules extractable from our ICLE training set of 490 texts, while the C&J parser produces 27,705. For reranking, we use only the C&J parser—since the parser stores these features during parsing, we can use them directly as classification features. On the ICLE training data, there are 6,230 features with frequency >10, and 19,659 with frequency >5.

4.3 Classifiers

For our experiments we used a maximum entropy (MaxEnt) machine learner, MegaM[4] (fifth release) by Hal Daumé III. (We also used an SVM for comparison, but the results were uniformly worse, and degraded more quickly as number of features increased, so we only report the MaxEnt results here). The classifier is tuned to obtain an optimal classification model.

4.4 Evaluation Methodology

Given our relatively small amount of data, we use k-fold cross-validation, choosing $k = 5$. While testing for statistical significance of classification results is often not carried out in NLP, we do so here because the quantity of data could raise questions about the certainty of any effect. In an encyclopedic survey of cross-validation in machine learning contexts, Refaeilzadeh et al. (2009) note that there is as yet no universal standard for testing of statistical significance; and that while more sophisticated techniques have been proposed, none is more widely accepted than a paired t-test over folds. We therefore use this paired t-test over folds, as formulated of Alpaydin

[3] Koppel et al. (2005) took all 258 texts per language from ICLE Version 1 and evaluated using 10-fold cross valiadation.

[4] MegaM is available on `http://www.cs.utah.edu/~hal/megam/`.

(2004). Under this cross-validation, 5 separate training feature sets are constructed, excluding the test fold; 3 folds are used for training, 1 fold for tuning and 1 fold for testing.

We also use a held-out test set for comparison, as it is well-known that cross-validation can overestimate prediction error (Hastie et al., 2009). We do not carry out significance testing here—with this held-out test set size ($n = 125$), two models would have to differ by a great deal to be significant. We only use it as a check on the effect of applying to completely new data.

5 Results

Table 1 presents the results for the three models individually under cross-validation. The first point to note is that PROD-RULE, under both parsers, is a substantial improvement over LEXICAL when (non-lexicalised) parse rules together with rules lexicalised with function words are used (rows marked with * in Table 1), with the largest difference as much as 77.75% for PROD-RULE[both]* ($n = all$) versus 64.29% for LEXICAL; these differences with respect to LEXICAL are statistically significant. (To give an idea, the paired t-test standard error for this largest difference is 2.52%.) In terms of error reduction, this is over 30%.

There appears to be no difference according to the parser used, regardless of their differing accuracy on the WSJ. Using the selection metric for PROD-RULE without rules lexicalised with function words produces results all around those for LEXICAL; using fewer reranking features is worse as the quality of RERANKING declines as feature cut-offs are raised.

Another, somewhat surprising point is that the RERANKING results are also generally around those of LEXICAL even though like PROD-RULE they are also using cross-sections of the parse tree. We consider there might be two possible reasons for this. The first is that the feature schemas used were originally chosen for the specific purpose of augmenting the performance of the Charniak parser; perhaps others might be more appropriate here. The second is that we selected only those instantiated feature schemas that occurred in the WSJ, and then applied them to ICLE. As the WSJ is filled with predominantly grammatical text, perhaps those that were not seen on the WSJ are precisely those that might indicate ungrammaticality. In contrast, the production rules of PROD-RULE were selected only from the ICLE training data.

Features	MaxEnt
LEXICAL ($n = 798$)	64.29
PROD-RULE[Stanford] ($n = 1000$)	65.72
PROD-RULE[Stanford]* ($n = 1000$)	74.08
PROD-RULE[Stanford]* ($n = all$)	74.49
PROD-RULE[C&J] ($n = 1000$)	62.25
PROD-RULE[C&J]* ($n = 1000$)	71.84
PROD-RULE[C&J]* ($n = all$)	71.63
PROD-RULE[both] ($n = 2000$)	67.96
PROD-RULE[both]* ($n = 2000$)	74.69
PROD-RULE[both]* ($n = all$)	77.75
RERANKING (all features)	67.96
RERANKING (>5 counts)	66.33
RERANKING (>10 counts)	64.90

Table 1: Classification results based on 5-fold cross validation with parse rules as syntactic features (accuracy %)

Features	MaxEnt
Lexical features ($n = 798$)	75.43
PROD-RULE[Stanford] ($n = 1000$)	74.29
PROD-RULE[Stanford]* ($n = 1000$)	79.43
PROD-RULE[Stanford]* ($n = all$)	78.86
PROD-RULE[C&J] ($n = 1000$)	73.71
PROD-RULE[C&J] ($n = 1000$)*	79.43
PROD-RULE[C&J] ($n = all$)*	80.00
PROD-RULE[both] ($n = 2000$)	77.71
PROD-RULE[both] ($n = 2000$)*	78.85
PROD-RULE[both] ($n = all$)*	80.00
RERANKING (all features)	77.14
RERANKING (>5 counts)	76.57
RERANKING (>10 counts)	75.43

Table 2: Classification results based on hold-out validation with parse rules as syntactic features (accuracy %)

Table 2 presents the results for the individual models on the held-out test set. The results are generally higher than for cross-validation—this is not surprising, as the texts are of the same type, but all the training data is used (rather than the $1-1/k$ proportion for cross-validation). Overall, the pattern is still the same, with PROD-RULE best, then RERANKING and LEXICAL broadly similar; as expected, no differences are significant with this smaller dataset. The gap has narrowed, but without significance test-

Features	MaxEnt
LEXICAL ($n = 798$)	64.29
LEXICAL + PROD-RULE[both] ($n = 2000$)	63.06
LEXICAL + PROD-RULE[both]* ($n = 2000$)	72.45
LEXICAL + PROD-RULE[both]* ($n = all$)	70.82
LEXICAL + RERANKING ($n = all$)	68.17

Table 3: Classification results based on 5-fold cross validation for combined models (accuracy %)

Features	MaxEnt
LEXICAL ($n = 798$)	75.43
LEXICAL + PROD-RULE[both] ($n = 2000$)	80.57
LEXICAL + PROD-RULE[both]* ($n = 2000$)	81.14
LEXICAL + PROD-RULE[both]* ($n = all$)	81.71
LEXICAL + RERANKING ($n = all$)	76.00

Table 4: Classification results based on hold-out validation for combined models (accuracy %)

ing it is difficult to say whether this is a genuine phenomenon. The accuracy rate for LEXICAL here is in line with Wong and Dras (2009); and given the smaller dataset and larger set of languages, also broadly in line with Koppel et al. (2005).

Tables 3 and 4 present results for model combinations. It can be seen that the model combinations do not produce results better than PROD-RULE alone. Combining all features (results not presented here) seems to degrade the overall performance even of the MegaM: perhaps we need to derive feature vectors more compactly than by feature concatenation.

6 Discussion

As illustrated in the confusion matrices (Table 5 for the PROD-RULE model, and Table 6 for the LEXICAL model), misclassifications occur largely in Spanish and Slavic languages, Bulgarian and Russian in particular. Unsurprisingly, Chinese is almost completely identified since it comes from a entirely different language family, Sino-Tibetan, as compared to the rest of the languages which are from the branches of the Indo-European family (with Japanese as the exception). Japanese and French also appear to be easily distinguished, which could probably be attributed to their word order or sentence structure which are, to some extent, quite different from English. Japanese is a 'subject-object-verb' language; and French, although having the same word order as English, heads of phrases in

	BL	CZ	FR	RU	SP	CN	JP
BL	[14]	6	2	3	-	-	-
CZ	1	[20]	-	3	1	-	-
FR	-	-	[25]	-	-	-	-
RU	1	4	3	[17]	-	-	-
SP	2	1	3	1	[18]	-	-
CN	-	-	-	-	-	[24]	1
JP	-	-	-	-	1	2	[22]

Table 5: Confusion matrix based on all non-lexicalised parse rules from both parsers on the held-out set (BL:Bulgarian, CZ:Czech, FR:French, RU:Russian, SP:Spanish, CN:Chinese, JP:Japanese)

	BL	CZ	FR	RU	SP	CN	JP
BL	[14]	3	2	4	2	-	-
CZ	6	[16]	-	2	1	-	-
FR	1	-	[24]	-	-	-	-
RU	3	2	3	[16]	1	-	-
SP	1	2	3	1	[17]	-	1
CN	-	-	-	-	-	[24]	1
JP	-	-	-	-	1	3	[21]

Table 6: Confusion matrix based on lexical features on the held-out set (BL:Bulgarian, CZ:Czech, FR:French, RU:Russian, SP:Spanish, CN:Chinese, JP:Japanese)

French typically come before modifiers as opposed to English. Overall, the PROD-RULE model results in fewer misclassifications compared to the LEXICAL model; there are mostly only incremental improvements for each language, with perhaps the exception of the reduction in confusion in the Slavic languages.

We looked at some of the data, to see what kind of syntactic substructure is useful in classifying native language. Although using feature selection with only 1000 features did not improve performance, the information gain ranking does identify particular constructions as characteristic of one of the languages, and so are useful for inspection.

A phenomenon that the literature has noted as occurring with Chinese speakers is that of the missing determiner.[5] This corresponds to a higher frequency of NP rules without determiners. These rules may be valid in other contexts, but are also used to describe ungrammatical constituents. One example is

[5]This does happen with native speakers of some other languages, such as Slavic ones, but not generally (from our knowledge of the literature) with native speakers of others, such as Romance ones.

Rules	Counts						
	BL	**CZ**	**FR**	**RU**	**SP**	**CN**	**JP**
NNP → <R>	0	0	3	0	0	67	0
: → -	55	51	23	39	10	9	4
PRN → -LRB- X -RRB-	0	1	7	2	0	42	0
SYM → *	0	1	7	3	1	42	0
: → :	30	39	58	46	47	11	6
X → SYM	0	2	7	4	4	42	6
NP → NNP NNP NNS	0	3	1	0	0	31	0
S → S : S .	36	34	53	39	41	5	9
PP → VBG PP	9	15	16	12	13	54	13
: → ...	16	13	39	11	24	1	3

Table 7: Top 10 rules for the Stanford parser according to Information Gain on the held-out set

```
(ROOT
  (S
    (NP
      (NP (DT The) (NN development))
      (PP (IN of)
        (NP (NN country) (NN park))))
    (VP (MD can)
      (ADVP (RB directly))
      (VP (VB elp)
        (S
          (VP (TO to)
            (VP (VB alleviate)
              (NP (NNS overcrowdedness)
                (CC and)
                (NN overpopulation))
              (PP (IN in)
                (NP (JJ urban)
                    (NN area)))))))))
    (. .)))
```

Figure 1: Parse from Chinese-speaking authors, illustrating missing determiner

```
(ROOT
  (S
    (PP (VBG According)
      (PP (TO to)
        (NP (NNP <R>))))
    (, ,)
    (NP
      (NP (NN burning))
      (PP (IN of)
        (NP (JJ plastic)
            (NN waste))))
    (VP (VBZ generates)
      (NP (JJ toxic)
          (NNS by-products)))
    (. .)))
```

Figure 2: Parse from Chinese-speaking authors, illustrating *according to*

NP → NN NN. In Figure 1 we give the parse (from the Stanford parser) of the sentence *The development of* ***country park*** *can directly elp to alleviate overcrowdedness and overpopulation in urban area*. The phrase *country park* should either have a determiner or be plural (in which case the appropriate rule would be NP → NN NNS). There is a similar phenomenon with *in urban area*, although this is an instance of the rule NP → JJ NN.

Another production rule that occurs typically—in fact, almost exclusively—in the texts of native Chinese speakers is PP → VBG PP (by the Stanford parser), which almost always corresponds to the phrase *according to*. In Figure 2 we give the parse of a short sentence (*According to <R>, burning of*

```
(S1
  (S
    (ADVP (RB Overall))
    (, ,)
    (NP (NNP cyber))
    (VP (VBD cafeis)
      (NP (DT a) (JJ good) (NN place))
      (PP (IN as)
          (NP (JJ recreational)
          (NNP centre)))
      (PP (IN with)
          (NP
            (NP
              (DT a) (NN bundle))
              (PP (IN of)
                (NP (JJ up-to-dated)
                  (NN information))))))
  (. .)))
```

Figure 3: Parse illustrating parser correction

plastic waste generates toxic by-products—<R>is an in-text citation that was removed in the preparation of ICLE) that illustrates this particular construction. It appears that speakers of Chinese frequently use this phrase as a translation of *gēn jù*. So in this case, what is identified is not the sort of error that is of interest to contrastive analysis, but just a particular construction that is characteristic of a certain native speaker's language, one that is perfectly grammatical but which is used relatively infrequently by others and has a slightly unusual analysis by the parser.

We had expected to see more rules that displayed obvious ungrammaticality, such as VP → DT IN. However, both parsers appear to be good at 'ignoring' errors, and producing relatively grammatical structures (albeit ones with different frequencies for different native languages). Figure 3 gives the C&J parse for *Overall, cyber **cafeis** a good place as recreational centre with a bundle of **up-to-dated** information*. The correction of *up-to-dated* rather than *up-to-date* is straightforward, but the simple typographical error of running together *cafe* and *is* leads to more complex problems for the parser. Nevertheless, the parser produces a solid grammatical tree, specifically assigning the category VBD to the compound *cafeis*. This appears to be because both the Stanford and C&J parsers have implicit linguistic constraints such as assumptions about heads; these are imposed even when the text does not provide evidence for them.

We also present in Table 7 the top 10 rules chosen under the IG feature selection for the Stanford parser on the held-out set. A number of these, and those ranked lower, are concerned with punctuation: these seem unlikely to be related to native language, but perhaps rather to how students of a particular language background are taught. Others are more typical of the sorts of example we illustrated above: PP → VBG PP, for example, is typically connected to the *according to* construction discussed in connection with Figure 2, and it can be seen that the dominant frequency count there is for native Chinese speakers (column 6 of the counts).

7 Conclusion

In this paper we have shown that, using cross-sections of parse trees, we can improve above an already good baseline in the task of native language identification. While we do not make any strong claims for the Contrastive Analysis Hypothesis, the usefulness of syntax in the context of this problem does provide some support.

The best features arising from the classification have been horizontal cross-sections of trees, rather than the more general discriminative parse reranking features that might have been expected to perform at least as well. This relatively poorer performance by the reranking features may be due to a number of factors, all of which could be investigated in future work. One is the use of feature schema instances that did not appear in the largely grammatical WSJ; another is the extension of feature schemas; and a third is the use of a parser that does not enforce linguistic constraints such as the Berkeley parser (Petrov et al., 2006).

Examining some of the substructures showed some errors that were expected; other constructions that were grammatical, but were just characteristic translations of constructions that were common in the native language; and a large number where grammatical errors were glossed over by the parser's linguistic constraints, suggesting another purpose for further work with the Berkeley parser. Overall, the use of these led to an error reduction in over 30%

in the cross-validation evaluation with significance testing.

Acknowledgments

The authors would like to acknowledge the support of ARC Linkage Grant LP0776267 and ARC Discovery Grant DP1095443, and thank the reviewers for useful feedback. Much gratitude is due to Mark Johnson for his guidance on the extraction of reranking features.

References

Ethem Alpaydin. 2004. *Introduction to Machine Learning*. MIT Press, Cambridge, MA, USA.

Harald Baayen, Hans van Halteren, and Fiona Tweedie. 1996. Outside the Cave of Shadows: Using Syntactic Annotation to Enhance Authorship Attribution. *Literary and Linguistic Computing*, 11(3):121–131.

Stephen Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180, Ann Arbor, Michigan.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

Michael Collins. 2000. Discriminative reranking for natural language processing. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00)*, Stanford, CA.

Stephen P. Corder. 1967. The significance of learners' errors. *International Review of Applied Linguistics in Language Teaching (IRAL)*, 5(4):161–170.

Dominique Estival, Tanja Gaustad, Son-Bao Pham, Will Radford, and Ben Hutchinson. 2007. Author profiling for English emails. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING)*, pages 263–272.

Ian Fette, Norman Sadeh, and Anthony Tomasic. 2007. Learning to detect phishing emails. In *Proceedings of the 16th International World Wide Web Conference*.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305.

Jennifer Foster, Joachim Wagner, and Josef van Genabith. 2008. Adapting a WSJ-trained parser to grammatically noisy text. In *Proceedings of ACL-08: HLT, Short Papers*, pages 221–224, Columbus, Ohio.

Julie Franck, Gabriella Vigliocco, and Janet Nicol. 2002. Subject-verb agreement errors in French and English: The role of syntactic hierarchy. *Language and Cognitive Processes*, 17(4):371–404.

Michael Gamon. 2004. Linguistic correlates of style: Authorship classification with deep linguistic analysis features. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 611–617.

Sylviane Granger and Stephanie Tyson. 1996. Connector usage in the English essay writing of native and non-native EFL speakers of English. *World Englishes*, 15(1):17–27.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. *International Corpus of Learner English (Version 2)*. Presses Universitaires de Louvain, Louvian-la-Neuve.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Mark Johnson and Ahmet Engin Ural. 2010. Reranking the Berkeley and Brown Parsers. In *Proceedings of Human Language Technologies: the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL-10)*, pages 665–668, Los Angeles, CA, USA, June.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic unification-based grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, MD.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. In *Intelligence and Security Informatics*, volume 3495 of *Lecture Notes in Computer Science*, pages 209–217. Springer-Verlag.

Robert Lado. 1957. *Linguistics Across Cultures: Applied Linguistics for Language Teachers*. University of Michigan Press, Ann Arbor, MI, US.

Brian MacWhinney and Elizabeth Bates. 1989. *The Crosslinguistic Study of Sentence Processing*. Cambridge University Press, New York, NY, USA.

Andrew Mutton, Mark Dras, Stephen Wan, and Robert Dale. 2007. GLEU: Automatic evaluation of

sentence-level fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 344–351, Prague, Czech Republic.

Steven Myers. 2007. Introduction to phishing. In Markus Jakobsson and Steven Myers, editors, *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Dominick Ng, Matthew Honnibal, and James R. Curran. 2010. Reranking a Wide-Coverage CCG Parser. In *Proceedings of Australasian Language Technology Association Workshop (ALTA'10)*, pages 90–98, Melbourne, Australia.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'06)*, pages 433–440, Sydney, Australia, July.

Manfred Pienemann. 1998. *Language Processing and Second Language Development: Processability Theory*. John Benjamins, Amsterdam, The Netherlands.

Payam Refaeilzadeh, Lei Tang, and Huan Liu. 2009. Cross-validation. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 532–538. Springer, US.

Jack C. Richards. 1971. A non-contrastive approach to error analysis. *ELT Journal*, 25(3):204–219.

Roumyana Slabakova. 2000. L1 transfer revisited: the L2 acquisition of telicity marking in English by Spanish and Bulgarian native speakers. *Linguistics*, 38(4):739–770.

Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhongyang Xiong, John Lee, and Chin-Yew Lin. 2007. Detecting erroneous sentences using automatically mined sequential patterns. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 81–88, Prague, Czech Republic.

Michael Swan and Bernard Smith, editors. 2001. *Learner English: A teacher's guide to interference and other problems*. Cambridge University Press, 2nd edition.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 353–358. Association for Computational Linguistics.

Oren Tsur and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 9–16.

Hans van Halteren. 2008. Source language markers in EUROPARL translations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 937–944.

Irena Vassileva. 1998. Who am I/how are we in academic writing? A contrastive analysis of authorial presence in English, German, French, Russian and Bulgarian. *International Journal of Applied Linguistics*, 8(2):163–185.

Garbriella Vigliocco, Brian Butterworth, and Merrill F. Garrett. 1996. Subject-verb agreement in Spanish and English: Differences in the role of conceptual constraints. *Cognition*, 61(3):261–298.

Joachim Wagner, Jennifer Foster, and Josef van Genabith. 2009. Judging grammaticality: Experiments in sentence classification. *CALICO Journal*, 26(3):474–490.

Richard Wardhaugh. 1970. The Contrastive Analysis Hypothesis. *TESOL Quarterly*, 4(2):123–130.

Sze-Meng Jojo Wong and Mark Dras. 2009. Contrastive analysis and native language identification. In *Proceedings of the Australasian Language Technology Association Workshop 2009*, pages 53–61, Sydney, Australia, December.

Sze-Meng Jojo Wong and Mark Dras. 2010. Parser features for sentence grammaticality classification. In *Proceedings of the Australasian Language Technology Association Workshop 2010*, pages 67–75, Melbourne, Australia, December.

Suying Yang and Yue-Yuan Huang. 2004. The impact of the absence of grammatical tense in L1 on the acquisition of the tense-aspect system in L2. *International Review of Applied Linguistics in Language Teaching (IRAL)*, 42(1):49–70.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, pages 412–420.

Rong Zheng, Yi Qin, Zan Huang, and Hsinchun Chen. 2003. Authorship analysis in cybercrime investigation. In *Intelligence and Security Informatics*, volume 2665 of *Lecture Notes in Computer Science*, pages 59–73. Springer-Verlag.

A Probabilistic Forest-to-String Model for Language Generation from Typed Lambda Calculus Expressions

Wei Lu and **Hwee Tou Ng**
Department of Computer Science
School of Computing
National University of Singapore
{luwei,nght}@comp.nus.edu.sg

Abstract

This paper describes a novel probabilistic approach for generating natural language sentences from their underlying semantics in the form of typed lambda calculus. The approach is built on top of a novel reduction-based weighted synchronous context free grammar formalism, which facilitates the transformation process from typed lambda calculus into natural language sentences. Sentences can then be generated based on such grammar rules with a log-linear model. To acquire such grammar rules automatically in an unsupervised manner, we also propose a novel approach with a generative model, which maps from sub-expressions of logical forms to word sequences in natural language sentences. Experiments on benchmark datasets for both English and Chinese generation tasks yield significant improvements over results obtained by two state-of-the-art machine translation models, in terms of both automatic metrics and human evaluation.

1 Introduction

This work focuses on the task of generating natural language sentences from their underlying meaning representations in the form of formal logical expressions (typed lambda calculus). Many early approaches to generation from logical forms make use of rule-based methods (Wang, 1980; Shieber et al., 1990), which concern surface realization (ordering and inflecting of words) but largely ignore lexical acquisition. Recent approaches start to employ corpus-based probabilistic methods, but many of them assume the underlying meaning representations are of specific forms such as variable-free tree-structured representations (Wong and Mooney, 2007a; Lu et al., 2009) or database entries (Angeli et al., 2010).

While these algorithms usually work well on specific semantic formalisms, it is unclear how well they could be applied to a different semantic formalism. In this work, we propose a general probabilistic model that performs generation from underlying formal semantics in the form of typed lambda calculus expressions (we refer to them as λ-*expressions* throughout this paper), where both lexical acquisition and surface realization are integrated in a single framework.

One natural proposal is to adopt a state-of-the-art statistical machine translation approach. However, unlike text to text translation, which has been extensively studied in the machine translation community, translating from logical forms into text presents additional challenges. Specifically, logical forms such as λ-expressions may have complex internal structures and variable dependencies across sub-expressions. Problems arise when performing automatic acquisition of a translation lexicon, as well as performing lexical selection and surface realization during generation.

In this work, we tackle these challenges by making the following contributions:

- *A novel forest-to-string generation algorithm*: Inspired by the work of Chiang (2007), we introduce a novel reduction-based weighted binary synchronous context-free grammar formalism for generation from logical forms (λ-expressions), which can then be integrated with a probabilistic forest-to-string generation algo-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1611–1622,
Edinburgh, Scotland, UK, July 27–31, 2011.

rithm.

- *A novel grammar induction algorithm*: To automatically induce such synchronous grammar rules, we propose a novel generative model that establishes phrasal correspondences between logical sub-expressions and natural language word sequences, by extending a previous model proposed for parsing natural language into meaning representations (Lu et al., 2008).

To our best knowledge, this is the first probabilistic model for generating sentences from the lambda calculus encodings of their underlying formal meaning representations, that concerns both surface realization and lexical acquisition. We demonstrate the effectiveness of our model in Section 5.

2 Related Work

The task of language generation from logical forms has a long history. Many early works do not rely on probabilistic approaches. Wang (1980) presented an approach for generation from an extended predicate logic formalism using hand-written rules. Shieber et al. (1990) presented a semantic head-driven approach for generation from logical forms based on rules written in Prolog. Shemtov (1996) presented a system for generation of multiple paraphrases from ambiguous logical forms. Langkilde (2000) presented a probabilistic model for generation from a packed forest meaning representation, without concerning lexical acquisition. Specifically, we are not aware of any prior work that handles both automatic unsupervised lexical acquisition and surface realization for generation from logical forms in a single framework.

Another line of research efforts focused on the task of language generation from other meaning representation formalisms. Wong and Mooney (2007a) as well as Chen and Mooney (2008) made use of synchronous grammars to transform a variable-free tree-structured meaning representation into sentences. Lu et al. (2009) presented a language generation model using the same meaning representation based on tree conditional random fields. Angeli et al. (2010) presented a domain-independent probabilistic approach for generation from database entries. All these models are probabilistic models.

Recently there are also substantial research efforts on the task of mapping natural language to meaning representations in various formalisms – the inverse task of language generation called *semantic parsing*. Examples include Zettlemoyer and Collins (2005; 2007; 2009), Kate and Mooney (2006), Wong and Mooney (2007b), Lu et al. (2008), Ge and Mooney (2009), as well as Kwiatkowski et al. (2010).

Of particular interest is our prior work Lu et al. (2008), in which we presented a joint generative process that produces a hybrid tree structure containing words, syntactic structures, and meaning representations, where the meaning representations are in a variable-free tree-structured form. One important property of the model in our prior work is that it induces a hybrid tree structure automatically in an unsupervised manner, which reveals the correspondences between natural language word sequences and semantic elements. We extend our prior model in the next section, so as to support λ-expressions. The model in turn serves as the basis for inducing the synchronous grammar rules later.

3 λ-Hybrid Tree

In Lu et al. (2008), a generative model was presented to model the process that jointly generates both natural language sentences and their underlying meaning representations of a variable-free tree-structured form. The model was defined over a *hybrid tree*, which consists of meaning representation tokens as internal nodes and natural language words as leaves. One limitation of the hybrid tree model is that it assumes a single fixed tree structure for the meaning representation. However, λ-expressions exhibit complex structures and variable dependencies, and thus it is not obvious how to represent them in a single tree structure.

In this section, we present a novel *λ-hybrid tree* model that provides the following extensions over the model of Lu et al. (2008):

1. The internal nodes of a meaning representation tree involve λ-expressions which are not necessarily of variable-free form;
2. The meaning representation has a packed forest representation, rather than a single deterministic tree structure.

3.1 Packed λ-Meaning Forest

We represent a λ-expression with a packed forest of meaning representation trees (called *λ-meaning for-*

est). Multiple different meaning representation trees (called λ-*meaning trees*) can be extracted from the same λ-meaning forest, but they all convey equivalent semantics via reductions, as discussed next.

Constructing a λ-meaning forest for a given λ-expression requires decomposition of a complete λ-expression into semantically complete and syntactically correct sub-expressions in a principled manner. This can be achieved with a process called higher order unification (Huet, 1975). The process was known to be very complex and was shown to be undecidable in unrestricted form (Huet, 1973). Recently a restricted form of higher order unification was applied to a semantic parsing task (Kwiatkowski et al., 2010). In this work, we employ a similar technique for building the λ-meaning forest.

For a given λ-expression e, our algorithm finds either two expressions h and f such that $(h\ f) \equiv e$, or three expressions h, f, and g such that $((h\ f)\ g) \equiv e$, where the symbol $\equiv$ is interpreted as α-*equivalent* after *reductions*[1] (Barendregt, 1985). We then build the λ-meaning forest based on the expressions h, f, and g. In practice, we develop a BUILDFOREST(e) procedure which recursively builds λ-forests by applying restricted higher-order unification rules on top of the λ-expression e. Each node of the λ-forest is called a λ-production, to which we will give more details in Section 3.2. For example, once a candidate triple (h, f, g) as in $((h\ f)\ g) \equiv e$ has been identified, the procedure creates a λ-forest with the root node being a λ-production involving h, and two sets of child λ-forests given by BUILDFOREST(f) and BUILDFOREST(g) respectively. For restricted higher-order unification, besides the similar assumptions made by Kwiatkowski et al. (2010), we also impose one additional assumption: limited free variable, which states that the expression h must contain no more than one free variable. Note that this process provides a semantically equivalent packed forest representation of the original λ-expression, without altering its semantics in any way.

For better readability, we introduce the symbol $\lhd$ as an alternative notation for functional application. In other words, $h \lhd f$ refers to $(h\ f)$ or $h(f)$, and $h \lhd f \lhd g$ refers to $((h\ f)\ g)$. For example, the expression $\lambda x.state(x) \wedge loc(boston, x)$ can be represented as the functional application form of $[\lambda f.\lambda x.f(x) \wedge loc(boston, x)] \lhd \lambda x.state(x)$.[2]

Such a packed forest representation contains exponentially many tree structures which all convey the same semantics. We believe such a semantic representation is more advantageous than the single fixed tree-structured representation. In fact, one could intuitively regard a different decomposition path as a different way of interpreting the same semantics. Thus, such a representation could potentially accommodate a wider range of natural language expressions, which all share the same semantics but with very different word choices, phrase orderings, and syntactic structures (like paraphrases). It may also alleviate the non-isomorphism issue that was commonly faced by researchers when mapping meaning representations and sentences (Wong and Mooney, 2007b). We will validate our belief later through experiments.

3.2 The Joint Generative Process

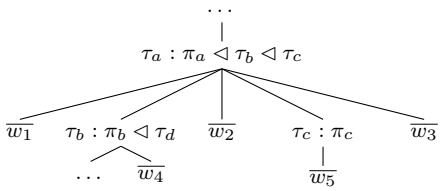


Figure 1: The joint generative process of both λ-meaning tree and its corresponding natural language sentence, which results in a λ-hybrid tree.

The generative process for a sentence together with its corresponding λ-meaning tree is illustrated in Figure 1, which results in a λ-*hybrid tree*. Internal nodes of a λ-hybrid tree are called λ-*productions*, which are building blocks of a λ-forest. Each λ-production in turn has at most two child λ-productions. A λ-production has the form $\tau_a : \pi_a \lhd \overline{\tau_b}$, where τ_a is the expected *type*[3] after type evaluation of the terms to its right, π_a is a λ-expression (serves as the functor), and $\overline{\tau_b}$ are types of the child λ-productions (as the arguments). The leave nodes

[1] In this work, for reductions, we consider α-conversions (changing bound variables) and β-conversions (applying functors to their arguments).

[2] Throughout this paper, we abuse this notation a bit by allowing the arguments to be types rather than actual expressions, such as $\lambda y.\lambda x.loc(y, x)) \lhd e$, which indicates that the functor $\lambda y.\lambda x.loc(y, x)$ expects an expression of type e to serve as its argument.

[3] This work considers basic types: e (entities) and t (truth values). It also allows function types, e.g., $\langle e, t\rangle$ is the type assigned to functions that map from entities to truth values.

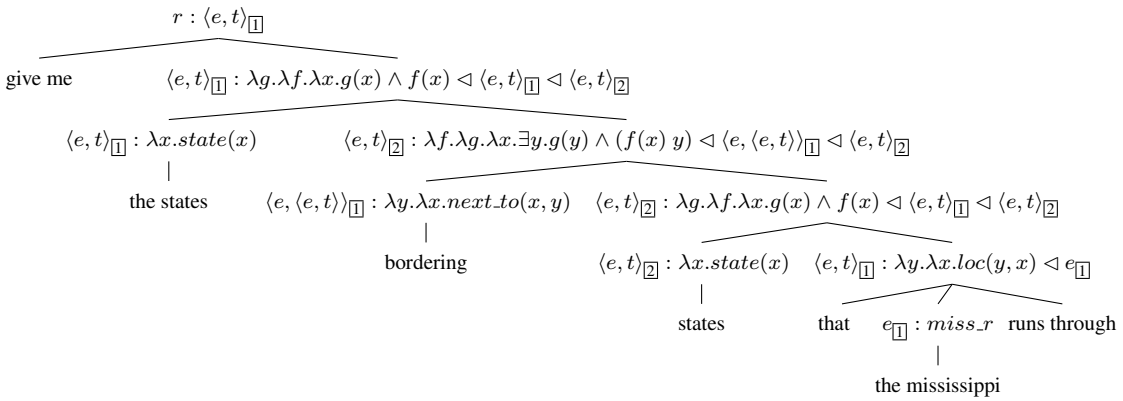


Figure 2: One example λ-hybrid tree for the sentence "give me the states bordering states that the mississippi runs through" together with its logical form "$\lambda x_0.state(x_0) \wedge \exists x_1.[loc(miss\_r, x_1) \wedge state(x_1) \wedge next\_to(x_1, x_0)]$".

$\overline{w}$ are contiguous word sequences. The model repeatedly generates λ-*hybrid sequences*, which consist of words intermixed with λ-productions, from each λ-production at different levels.

Consider part of the example λ-hybrid tree in Figure 2. The probability associated with generation of the subtree that spans the sub-sentence "that the mississippi runs through" can be written as:

$$P\Big(\lambda x.loc(miss\_r, x), \text{that the mississippi runs through}\Big)$$
$$= \phi(m \rightarrow \overline{\mathbf{w}\mathcal{Y}\mathbf{w}}|p_1) \times \psi(\text{that } e_{\boxed{1}} \text{ runs through}|p_1)$$
$$\times \rho(p_2|p_1, \arg_1) \times \phi(m \rightarrow \overline{\mathbf{w}}|p_2) \times \psi(\text{the mississippi}|p_2)$$

where $p_1 = \langle e,t\rangle : \lambda y.\lambda x.loc(y,x) \lhd e_{\boxed{1}}$, and $p_2 = e : miss\_r$.

Following the work of Lu et al. (2008), the generative process involves three types of parameters $\overline{\theta} = \{\phi, \psi, \rho\}$: 1) pattern parameters ϕ, which model in what way the words and child λ-productions are intermixed; 2) emission parameters ψ, which model the generation process of words from λ-productions, where either a unigram or a bigram assumption can be made (Lu et al., 2008); and 3) meaning representation (MR) model parameters ρ, which model the generation process from one λ-production to its child λ-productions. An analogous inside-outside algorithm (Baker, 1979) used there is employed here. Since we allow a packed λ-meaning forest representation rather than a fixed tree structure, the MR model parameters ρ in this work should be estimated with the inside-outside algorithm as well, rather than being estimated directly from the training data by simple counting, as was done in Lu et al. (2008).

4 The Language Generation Algorithm

Now we present the algorithm for language generation. We introduce the grammar first, followed by the features we use. Next, we present the method for grammar induction, and then discuss the decoder.

4.1 The Grammar

We use a weighted synchronous context free grammar (SCFG) (Aho and Ullman, 1969), which was previously used in Chiang (2007) for hierarchical phrase-based machine translation. The grammar is defined as follows:

$$\tau \rightarrow \quad \langle p_\lambda\ , h_w, \sim \rangle \qquad (1)$$

where τ is the type associated with the λ-production p_λ[4], and h_w is a sequence consisting of natural language words intermixed with types. The symbol $\sim$ denotes the one-to-one correspondence between nonterminal occurrences (i.e., in this case types of λ-expressions) in both p_λ and h_w.

We allow a maximum of two nonterminal symbols in each synchronous rule, as was also assumed in Chiang (2007), which makes the grammar a binary SCFG. Two example rules are:

$$\langle e,t\rangle \rightarrow \Big\langle \lambda y.\lambda x.loc(y,x) \lhd e_{\boxed{1}}\ ,\ \text{that } e_{\boxed{1}} \text{ runs through} \Big\rangle$$
$$e \rightarrow \Big\langle miss\_r,\ \text{the mississippi} \Big\rangle$$

where the boxed indices give the correspondences between nonterminals.

A derivation with the above two synchronous rules results in the following λ-expression paired with its natural language counterpart:

[4]Since type is already indicated by τ, we avoid redundancy by omitting it when writing p_λ, without loss of information.

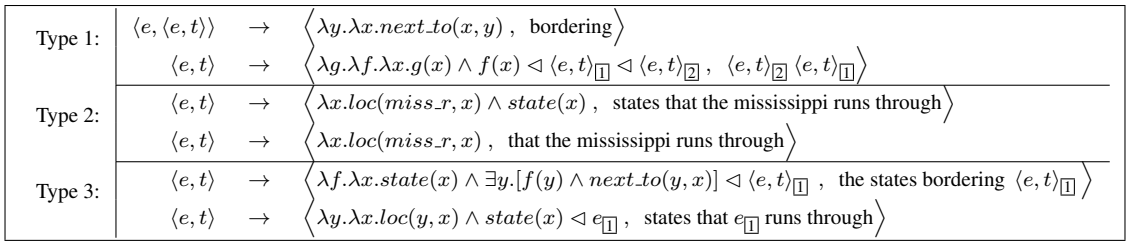


Figure 3: Example synchronous rules that can be extracted from the λ-hybrid tree of Figure 2.

$$\langle e, t\rangle \rightarrow \left\langle \lambda x.loc(miss\_r, x)\ ,\ \text{that the mississippi runs through} \right\rangle$$

where the source side λ-expression is constructed from the application $\lambda y.\lambda x.loc(y, x) \lhd miss\_r$ followed by a reduction (β-conversion). Assuming the λ-expression to be translated is $\lambda x.loc(miss\_r, x)$, the above rule in fact gives one candidate translation "that the mississippi runs through".

4.2 Features

Following the work of Chiang (2007), we assign scores to derivations with a log-linear model, which are essentially weighted products of feature values.

For generality, we only consider the following four simple features in this work:

1. $\tilde{p}(h_w|p_\lambda)$: the relative frequency estimate of a hybrid sequence h_w given the λ-production p_λ;
2. $\tilde{p}(p_\lambda|h_w, \tau)$: the relative frequency estimate of a λ-production p_λ given the phrase h_w and the type τ;
3. $\exp(-wc(h_w))$: the number of words generated, where $wc(h_w)$ refers to the number of words in h_w (i.e., word penalty); and
4. $p_{LM}(\hat{s})$: the language model score of the generated sentence $\hat{s}$.

The first three features, which are also widely used in state-of-the-art machine translation models (Koehn et al., 2003; Chiang, 2007), are rule-specific and thus can be computed before decoding. The last feature is computed during the decoding phase in combination with the sibling rules used.

We score a derivation D with a log-linear model:

$$\mathbf{w}(D) = \left(\prod_{r \in D} \prod_i f_i(r)^{w_i} \right) \times p_{LM}(\hat{s})^{w_{LM}} \qquad (2)$$

where $r \in D$ refers to a rule r that appears in the derivation D, $\hat{s}$ is the target side (sentence) associated with the derivation D, and f_i is a rule-specific feature (one of features 1–3 above) which is weighted with w_i. The language model feature is weighted with w_{LM}.

Once the feature values are computed, our goal is to find the optimal weight vector $\bar{w}^*$ that maximizes a certain evaluation metric when used for decoding, as we will discuss in Section 4.4.

Following popular approaches to learning feature weights in the machine translation community (Och and Ney, 2004; Chiang, 2005), we use the minimum error rate training (MERT) (Och, 2003) algorithm to learn the feature weights that directly optimize certain automatic evaluation metric. Specifically, the Z-MERT (Zaidan, 2009) implementation of the algorithm is used in this work.

4.3 Grammar Induction

Automatic induction of the grammar rules as described above from training data (which consists of pairs of λ-expressions and natural language sentences) is a challenging task. Current state-of-the-art string-based translation systems (Koehn et al., 2003; Chiang, 2005; Galley and Manning, 2010) typically begin with a word-aligned corpus to construct phrasal correspondences. Word-alignment information can be estimated from alignment models, such as the IBM alignment models (Brown et al., 1993) and HMM-based alignment models (Vogel et al., 1996; Liang et al., 2006). However, unlike texts, logical forms have complex internal structures and variable dependencies across sub-expressions. It is not obvious how to establish alignments between logical terms and texts with such alignment models.

Fortunately, the generative model for λ-hybrid tree introduced in Section 3 explicitly models the mappings from λ-sub-expressions to (possibly discontiguous) word sequences with a joint generative process. This motivates us to extract grammar rules from the λ-hybrid trees. Thus, we first find the Viterbi λ-hybrid trees for all training instances,

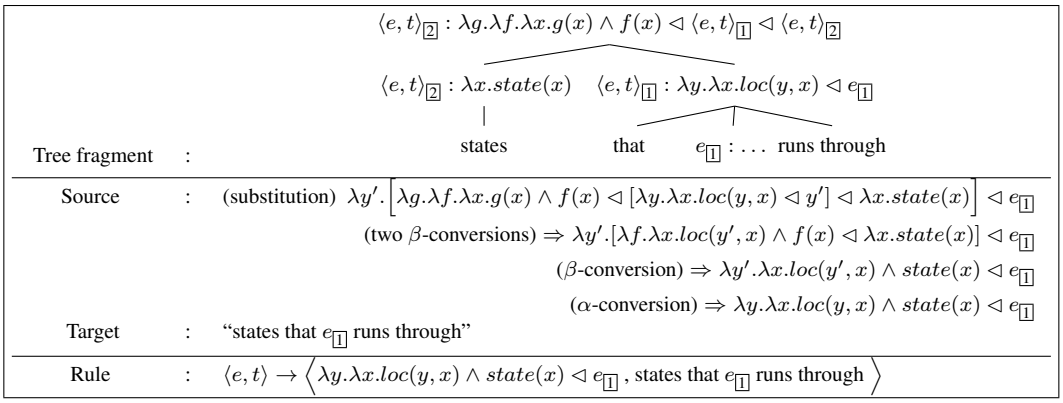


Figure 4: Construction of a two-level λ-hybrid sequence rule via substitution and reductions from a tree fragment. Note that the subtree rooted by $e_{\boxed{1}} : miss\_r$ gets "abstracted" by its type e. The auxiliary variable y' of type e is thus introduced to facilitate the construction process.

based on the learned parameters of the generative λ-hybrid tree model.

Next, we extract grammar rules on top of these λ-hybrid trees. Specifically, we extract the following three types of synchronous grammar rules, with examples given in Figure 3:

1. λ-hybrid sequence rules: They are the conventional rules constructed from one λ-production and its corresponding λ-hybrid sequence.
2. Subtree rules: These rules are constructed from a complete subtree of the λ-hybrid tree. Each rule provides a mapping between a complete sub-expression and a contiguous sub-sentence.
3. Two-level λ-hybrid sequence rules: These rules are constructed from a tree fragment with one of its grandchild subtrees (the subtree rooted by one of its grandchild nodes) being abstracted with its type only. These rules are constructed via substitution and reductions.
   Figure 4 gives an example based on a tree fragment of the λ-hybrid tree in Figure 2. Note that the first step makes use of the auxiliary variable y' of type e to represent the grandchild subtree. $\lambda y'$ is introduced so as to allow any λ-expression of type e serving as this expression's argument to replace y'. In fact, if the semantics conveyed by the grandchild subtree serves as its argument, we will obtain the exact complete semantics of the current subtree. As we can see, the resulting rule is more general, and is able to capture longer structural dependencies. Such rules are thus potentially more useful.

The overall algorithm for learning the grammar rules is sketched in Figure 5.

4.4 Decoding

Our goal in decoding is to find the most probable sentence $\hat{s}$ for a given λ-expression e:

$$\hat{s} = s\left(\arg\max_{D \text{ s.t. } e(D) \equiv e} \mathbf{w}(D)\right) \quad (3)$$

where $e(D)$ refers to the source side (λ-expression) of the derivation D, and $s(D)$ refers to the target side (natural language sentence) of D.

A conventional CKY-style decoder as used by Chiang (2007) is not applicable to this work since the source side does not exhibit a linear structure. As discussed in Section 3.1, λ-expressions are represented as packed λ-meaning forests. Thus, in this work, we make use of a bottom-up dynamic programming chart-parsing algorithm that works directly on translating forest nodes into target natural language words. The algorithm is similar to that of Langkilde (2000) for generation from an underlying packed semantic forest. Language models are incorporated when scoring the n-best candidates at each forest node, where the cube-pruning algorithm of Chiang (2007) is used. In order to accommodate type 2 and type 3 rules as discussed in Section 4.3, whose source side λ-productions are not present in the nodes of the original λ-meaning forest, new λ-productions are created (via substitution and reductions) and attached to the original λ-meaning forest.

Procedures

- $f \leftarrow \text{BUILDFOREST}(e)$
  It takes in a λ-expression e and outputs its λ-meaning forest f. (Sec. 3.1)
- $\bar{\theta} \leftarrow \text{TRAINGENMODEL}(\mathbf{f}, \mathbf{s})$
  It takes in λ-meaning forest-sentence pairs $(\mathbf{f}, \mathbf{s})$, performs EM training of the generative model, and outputs the parameters $\bar{\theta}$. (Sec. 3.2)
- $h \leftarrow \text{FINDHYBRIDTREE}(f, s, \bar{\theta})$
  It finds the most probable λ-hybrid tree h containing the given f-s pair, under the generative model parameters $\bar{\theta}$. (Sec. 4.3)
- $\Gamma_h \leftarrow \text{EXTRACTRULES}(h)$
  It takes in a λ-hybrid tree h, and extracts a set of grammar rules Γ_h out of it. (Sec. 4.3)

Algorithm

1. Inputs and initializations:
   - A training set $(\mathbf{e}, \mathbf{s})$, an empty rule set $\Gamma = \emptyset$
2. Learn the grammar:
   - For each $e_i \in \mathbf{e}$, find its λ-meaning forest: $f_i = \text{BUILDFOREST}(e_i)$. This gives the set $(\mathbf{f}, \mathbf{s})$.
   - Learn the generative model parameter : $\bar{\theta}^* = \text{TRAINGENMODEL}(\mathbf{f}, \mathbf{s})$.
   - For each $(f_i, s_i) \in (\mathbf{f}, \mathbf{s})$, find the most probable λ-hybrid tree h_i, and then extract the grammar rules from it:
     $h_i = \text{FINDHYBRIDTREE}(f_i, s_i, \bar{\theta}^*)$
     $\Gamma = \Gamma \cup \text{EXTRACTRULES}(h_i)$
3. Output the learned grammar rule set Γ.

Figure 5: The algorithm for learning the grammar rules

5 Experiments

For experiments, we evaluated on the GEOQUERY dataset, which consists of 880 queries on U.S. geography. The dataset was manually labeled with λ-expressions as their semantics in Zettlemoyer and Collins (2005). It was used in many previous research efforts on semantic parsing (Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010). The original dataset was annotated with English sentences only. In order to assess the generation performance across different languages, in our work the entire dataset was also manually annotated with Chinese by a native Chinese speaker with linguistics background[5].

For all the experiments we present in this section, we use the same split as that of Kwiatkowski et al. (2010), where 280 instances are used for testing, and the remaining instances are used for learning. We further split the learning set into two portions, where 500 instances are used for training the models, which includes induction of grammar rules, training a language model, and computing feature values, and the remaining 100 instances are used for tuning the feature weights.

As we have mentioned earlier, we are not aware of any previous work that performs generation from formal logical forms that concerns both lexical acquisition and surface realization. The recent work by Angeli et al. (2010) presented a generation system from database records with an additional focus on content selection (selection of records and their subfields for generation). It is not obvious how to adopt their algorithm in our context where content selection is not required but the more complex logical semantic representation is used as input. Other earlier approaches such as the work of Wang (1980) and Shieber et al. (1990) made use of rule-based approaches without automatic lexical acquisition.

We thus compare our system against two state-of-the-art machine translation systems: a phrase-based translation system, implemented in the Moses toolkit (Koehn et al., 2007)[6], and a hierarchical phrase-based translation system, implemented in the Joshua toolkit (Li et al., 2009), which is a reimplementation of the original Hiero system (Chiang, 2005; Chiang, 2007). The state-of-the-art unsupervised Berkeley aligner (Liang et al., 2006) with default setting is used to construct word alignments. We train a trigram language model with modified Kneser-Ney smoothing (Chen and Goodman, 1996) from the training dataset using the SRILM toolkit (Stolcke, 2002), and use the same language model for all three systems. We use an n-best list of size 100 for all three systems when performing MERT.

5.1 Automatic Evaluation

For automatic evaluation, we measure the original IBM BLEU score (Papineni et al., 2002) (4-gram precision with brevity penalty) and the TER score (Snover et al., 2006) (the amount of edits required to change a system output into the reference)[7]. Note that TER measures the translation error rate, thus a

[5]The annotator created annotations with both λ-expressions and corresponding English sentences available as references.

[6]We used the default settings, and enabled the default lexicalized reordering model, which yielded better performance.

[7]We used tercom version 0.7.25 with the default settings.

smaller score indicates a better result. For clarity, we report 1−TER scores. Following the tuning procedure as conducted in Galley and Manning (2010), we perform MERT using BLEU as the metric.

We compare our model against state-of-the-art statistical machine translation systems. As a baseline, we first conduct an experiment with the following naive approach: we treat the λ-expressions as plain texts. All the bound variables (e.g., x in $\lambda x.state(x)$) which do not convey semantics are removed, but free variables (e.g., $state$ in $\lambda x.state(x)$) which might convey semantics are left intact. Quantifiers and logical connectives are also left intact. While this naive approach might not appear very sensible, we merely want to treat it as our simplest baseline.

Alternatively, analogous to the work of Wong and Mooney (2007a), we could first parse the λ-expressions into binary tree structures with a deterministic procedure, and then linearize the tree structure as a sequence. Since there exists different ways to linearize a binary tree, we consider preorder, inorder, and postorder traversal of the trees, and linearize them in these three different ways.

As for our system, during the grammar learning phase, we initialize the generative model parameters with output from the IBM alignment model 1 (Brown et al., 1993)[8], and run the λ-hybrid tree generative model with the unigram emission assumption for 10 iterations, followed by another 10 iterations with the bigram assumption. Grammar rules are then extracted based on the λ-hybrid trees obtained from such learned generative model parameters.

Since MERT is prone to search errors, we run each experiment 5 times with randomly initialized feature weights, and report the averaged scores. Experimental results for both English and Chinese are presented in Table 1. As we can observe, the way that a meaning representation tree is linearized has a significant impact on the translation performance. Interestingly, for both Moses and Joshua, the preorder setting yields the best performance for English, whereas it is inorder that yields the best performance for Chinese. This is perhaps due to the fact that Chinese presents a very different syntactic structure and word ordering from English.

[8]We assume word unigrams are generated from free variables, quantifiers, and logical connectives in IBM model 1.

Our system, on the other hand, employs a packed forest representation for λ-expressions. Therefore, it eliminates the ordering constraint by encompassing exponentially many possible tree structures during both the alignment and decoding stage. As a result, our system obtains significant improvements in both BLEU and 1−TER using the significance test under the paired bootstrap resampling method of Koehn (2004). We obtain $p < 0.01$ for all cases, except when comparing against Joshua-preorder for English, where we obtain $p < 0.05$ for both metrics.

		English		Chinese	
		BLEU	1−TER	BLEU	1−TER
Moses	text	48.93	61.08	43.23	51.71
	preorder	*51.13*	*63.73*	42.08	50.43
	inorder	46.72	57.59	*48.03*	*55.29*
	postorder	44.30	55.05	46.36	54.59
Joshua	text	37.40	48.97	36.60	46.20
	preorder	*51.40*	*64.69*	40.05	49.70
	inorder	40.31	50.47	*48.32*	*54.64*
	postorder	31.10	42.44	41.31	49.71
This work (t)		**54.58**	**67.65**	**55.11**	**63.77**
(t) w/o type 2 rules		53.77	66.43	54.30	62.49
(t) w/o type 3 rules		53.68	66.17	50.96	60.13

Table 1: Performance on generating English and Chinese from λ-expressions with automatic evaluation metrics (we report percentage scores).

5.2 Human Evaluation

We also conducted human evaluation with 5 evaluators each on English and Chinese. We randomly selected about 50% (139) test instances and obtained output sentences from the three systems. Moses and Joshua were run with the top-performing settings in terms of automatic metrics (i.e., preorder for English and inorder for Chinese). Following Angeli et al. (2010), evaluators are instructed to give scores based on language fluency and semantic correctness, on the following scale:

Score	Language Fluency	Semantic Correctness
5	Flawless	Perfect
4	Good	Near Perfect
3	Non-native	Minor Errors
2	Disfluent	Major Errors
1	Gibberish	Completely Wrong

For each test instance, we first randomly shuffled the output sentences of the three systems, and presented them together with the correct reference to the evaluators. The evaluators were then asked to score all the output sentences at once. This evaluation process not only ensures that the annotators have no access to which system generated the out-

English	Judge E1		Judge E2		Judge E3		Judge E4		Judge E5		Average	
	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM
Moses preorder	4.56	4.57	4.58	4.54	4.52	4.52	4.48	4.14	4.28	4.22	4.48 ± 0.12	4.40 ± 0.20
Joshua preorder	4.50	4.43	4.49	4.29	4.44	4.36	4.46	4.04	4.12	4.06	4.40 ± 0.16	4.24 ± 0.18
This work	**4.76**	**4.73**	**4.73**	**4.70**	**4.68**	**4.60**	**4.64**	**4.37**	**4.49**	**4.44**	**4.66** ± 0.10	**4.57** ± 0.16
Chinese	Judge C1		Judge C2		Judge C3		Judge C4		Judge C5		Average	
	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM
Moses inorder	4.38	4.22	3.95	3.99	4.01	3.80	4.27	4.19	4.09	4.01	4.14 ± 0.18	4.04 ± 0.17
Joshua inorder	4.32	4.04	3.74	3.91	3.76	3.55	4.21	4.04	3.96	3.97	4.00 ± 0.26	3.90 ± 0.21
This work	**4.61**	**4.47**	**4.53**	**4.43**	**4.50**	**4.31**	**4.71**	**4.55**	**4.57**	**4.32**	**4.59** ± 0.08	**4.42** ± 0.10

Table 2: Human evaluation results on English and Chinese generation. FLU: language fluency; SEM: semantic correctness.

put, but also minimizes bias associated with scoring different outputs for the same input. The detailed and averaged results (with one standard deviation) for human evaluation are presented in Table 2 for English and Chinese respectively. For both languages, our system achieves a significant improvement over Moses and Joshua ($p < 0.01$ with paired t-tests), in terms of both language fluency and semantic correctness. This set of results is important, as it demonstrates that our system produces more fluent texts with more accurate semantics when perceived by real humans.

5.3 Additional Experiments

We also performed the following additional experiments. First, we attempted to increase the number of EM iterations (to 100) when training the model with the bigram assumption, so as to assess the effect of the number of EM iterations on the final generation performance. We observed similar performance. Second, in order to assess the importance of the two types of novel rules – subtree rules (type 2) and two-level λ-hybrid sequence rules (type 3), we also conducted experiments without these rules for generation. Experiments show that these two types of rules are important. Specifically, type 3 rules, which are able to capture longer structural dependencies, are of particular importance for generating Chinese. Detailed results for these additional experiments are presented in Table 1.

5.4 Experiments on Variable-free Meaning Representations

Finally, we also assess the effectiveness of our model on an alternative meaning representation formalism in the form of variable-free tree structures. Specifically, we tested on the ROBOCUP dataset (Kuhlmann et al., 2004), which consists of 300 English instructions for coaching robots for soccer games, and a variable-free version of the GEOQUERY dataset. These are the standard datasets used in the generation tasks of Wong and Mooney (2007a) and Lu et al. (2009). Similar to the technique introduced in Kwiatkowski et al. (2010), our proposed algorithm could still be applied to such datasets by writing the tree-structured representations as function-arguments forms. The higher order unification-based decomposition algorithm could be applied on top of such forms accordingly. For example, $midfield(opp) \equiv \lambda x.midfield(x) \lhd opp$. See Kwiatkowski et al. (2010) for more details. However, since such forms present monotonous structures, and thus give less alternative options in the higher-order unification-based decomposition process, it prevents the algorithm from creating many disjunctive nodes in the packed forest. It is thus hypothesized that the advantages of the packed forest representation could not be fully exploited with such a meaning representation formalism.

Following previous works, we performed 4 runs of 10-fold cross validation based on the same split as that of Wong and Mooney (2007a) and Lu et al. (2009), and measured standard BLEU percentage and NIST (Doddington, 2002) scores. For experimentation on each fold, we trained a trigram language model on the training data of that fold, and randomly selected 70% of the training data for grammar induction, with the remaining 30% for learning of the feature weights using MERT. Next, we performed grammar induction with the complete training data of that fold, and used the learned feature weights for decoding of the test instances. The averaged results are shown in Table 3. Our approach outperforms the previous system WASP^{-1}++ (Wong and Mooney, 2007a) significantly, and achieves comparable or slightly better performance as compared to Lu et al. (2009). This set of results is particularly striking. We note

Variable-present dataset	
λ-expression :	$argmax(x, river(x) \wedge \exists y.[state(y) \wedge next\_to(y, india\_s) \wedge loc(x, y)], len(x))$
Reference :	what is the longest river that flows through a state that borders indiana
Moses :	what is the states that border long indiana
Joshua :	what is the longest river surrounding states border indiana
This work :	what is the longest river in the states that border indiana
λ-expression :	$density(\imath x.loc(argmax(y, loc(y, usa\_co) \wedge river(y), size(y)), x) \wedge state(x))$
Reference :	which is the density of the state that the largest river in the united states runs through
Moses :	what is the population density in lie on the state with the smallest state in the us
Joshua :	what is the population density of states lie on the smallest state in the us
This work :	what is the population density of the state with the largest river in the us
Variable-free datasets	
λ-expression :	$population(largest\_one\_density(state\_all))$
Reference :	what is the population of the state with the highest population density
This work :	how many people live in the state with the largest population density
λ-expression :	$rule(and(bpos(from\_goal\_line(our, jnum(n0.0, n32.0))), not(bpos(left(penalty\_area(our))))),$- $dont(player\_our(n3), intercept))$
Reference :	player 3 should not intercept the ball if the ball is within 32 meters of our goal line and not in our left penalty area
This work :	if the ball is within 32 meters from our goal line and not on the left side of our penalty area then player 3 should not intercept it

Figure 6: Sample English outputs for various datasets. For the variable-present dataset, we also show outputs from Moses and Joshua.

that the algorithm of Lu et al. (2009) is capable of modeling dependencies over phrases, which gives global optimization over the sentence generated, and works by building conditional random fields (Lafferty et al., 2001) over trees. But the algorithm of Lu et al. (2009) is also limited to handling tree-structured meaning representation, and is therefore unable to accept inputs such as the variable version of λ-expressions. Our algorithm works well by introducing additional new types of synchronous rules that are able to capture longer range dependencies. WASP^{-1}++, on the other hand, also makes use of a synchronous parsing-based statistical machine translation approach. Their system, however, requires linearization of the tree structure for both alignment and translation. In contrast, our model directly performs alignment and translation from a packed forest representation to a sentence. As a result, though WASP^{-1}++ made use of additional features (lexical weights), our system yielded better performance. Sample English output sentences are given in Figure 6.

	Robocup		Geoquery	
	BLEU	NIST	BLEU	NIST
WASP^{-1}++	60.22	6.8976	53.70	6.4808
Lu et al. (2009)	62.20	6.9845	57.33	**6.7459**
This work	**62.45**	**7.0011**	**57.62**	6.6867

Table 3: Performance on variable-free representations

6 Conclusions and Future Work

In this work, we presented a novel algorithm for generating natural language sentences from their underlying semantics in the form of typed lambda calculus. We tackled the problem by introducing a novel reduction-based weighted synchronous context-free grammar formalism, which allows sentence generation with a log-linear model. In addition, we proposed a novel generative model that jointly generates lambda calculus expressions and natural language sentences. The model is then used for automatic grammar induction. Empirical results show that our model outperforms state-of-the-art machine translation models, for both English and Chinese, in terms of both automatic and human evaluation. Furthermore, we have demonstrated that the model can also effectively handle inputs with a variable-free version of meaning representation.

We believe the algorithm used for inducing the reduction-based synchronous grammar rules may find applications in other research problems, such as statistical machine translation and phrasal synchronous grammar induction. We are interested in exploring further along such directions in the future.

Acknowledgments

This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore. We would like to thank Tom Kwiatkowski and Luke Zettlemoyer for sharing their dataset, and Omar F. Zaidan for his help with Z-MERT.

References

A.V. Aho and J.D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56.

G. Angeli, P. Liang, and D. Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proc. EMNLP*, pages 502–512.

J. K. Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65:S132.

H. P. Barendregt. 1985. *The Lambda Calculus, Its Syntax and Semantics (Studies in Logic and the Foundations of Mathematics, Volume 103). Revised Edition*. North-Holland.

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

S. F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. ACL*, pages 310–318.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proc. ICML*, pages 128–135.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33:201–228.

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. HLT*, pages 138–145.

M. Galley and C. D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Proc. HLT/NAACL*, pages 966–974.

R. Ge and R. J. Mooney. 2009. Learning a compositional semantic parser using an existing syntactic parser. In *Proc. ACL/IJCNLP*, pages 611–619.

G. P. Huet. 1973. The undecidability of unification in third order logic. *Information and Control*, 22(3):257–267.

G. P. Huet. 1975. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1(1):27–57.

R. J. Kate and R. J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proc. COLING/ACL*, pages 913–920.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL/HLT*, pages 48–54.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proc. ACL (Demonstration Sessions)*, pages 177–180.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*, pages 388–395.

G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik. 2004. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *Proc. AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proc. EMNLP*, pages 1223–1233.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, pages 282–289.

I. Langkilde. 2000. Forest-based statistical sentence generation. In *Proc. NAACL*, pages 170–177.

Z. Li, C. Callison-Burch, C. Dyer, J. Ganitkevitch, S. Khudanpur, L. Schwartz, W. N. G. Thornton, J. Weese, and O. F. Zaidan. 2009. Joshua: an open source toolkit for parsing-based machine translation. In *Proc. WMT*, pages 135–139.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. HLT/NAACL*, pages 104–111.

W. Lu, H. T. Ng, W. S. Lee, and L. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proc. EMNLP*, pages 783–792.

W. Lu, H. T. Ng, and W. S. Lee. 2009. Natural language generation with tree conditional random fields. In *Proc. EMNLP*, pages 400–409.

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

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.

H. Shemtov. 1996. Generation of paraphrases from ambiguous logical forms. In *Proc. COLING*, pages 919–924.

S. M. Shieber, G. van Noord, F. C. N. Pereira, and R. C. Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16(1):30–42.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. AMTA*, pages 223–231.

A. Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proc. ICSLP*, pages 901–904.

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

J. Wang. 1980. On computational sentence generation from logical form. In *Proc. COLING*, pages 405–411.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. HLT/NAACL*, pages 439–446.

Y. W. Wong and R. J. Mooney. 2007a. Generation by inverting a semantic parser that uses statistical machine translation. In *Proc. NAACL/HLT*, pages 172–179.

Y. W. Wong and R. J. Mooney. 2007b. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. ACL*, pages 960–967.

O. F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. UAI*, pages 658–666.

L. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. EMNLP-CoNLL*, pages 678–687.

L. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proc. ACL/IJCNLP*, pages 976–984.

Author Index